Similarity Search and Mining in Uncertain Spatial and Spatio-Temporal Databases

Andreas Züfle

München 2013

Similarity Search and Mining in Uncertain Spatial and Spatio-Temporal Databases

Andreas Züfle

Dissertation
an der Fakultät für Mathematik, Informatik und Statistik
der Ludwig–Maximilians–Universität
München

vorgelegt von
Andreas Züfle
aus München

München, den 22.05.2013

Erstgutachter: Prof. Dr. Hans-Peter Kriegel
Zweitgutachter: Prof. Dr. Nikos Mamoulis
Tag der mündlichen Prüfung: 23.08.2013

Eidesstattliche Versicherung

(Siehe Promotionsordnung vom 12.07.11, § 8, Abs. 2 Pkt. .5.)

Hiermit erkläre ich an Eidesstatt, dass die Dissertation von mir selbstständig, ohne unerlaubte Beihilfe angefertigt ist.

Name, Vorname

....

Ort, Datum Unterschrift Doktorand/in

Contents

List of Figures

List of Tables

Abstract

Both the current trends in technology such as smart phones, general mobile devices, stationary sensors and satellites as well as a new user mentality of utilizing this technology to voluntarily share information produce a huge flood of geo-spatial and geo-spatio-temporal data. This data flood provides a tremendous potential of discovering new and possibly useful knowledge. In addition to the fact that measurements are imprecise, due to the physical limitation of the devices, some form of interpolation is needed in-between discrete time instances. From a complementary perspective - to reduce the communication and bandwidth utilization, along with the storage requirements, often the data is subjected to a reduction, thereby eliminating some of the known/recorded values. These issues introduce the notion of uncertainty in the context of spatio-temporal data management - an aspect raising an imminent need for scalable and flexible data management.

The main scope of this thesis is to develop effective and efficient techniques for similarity search and data mining in uncertain spatial and spatio-temporal data. In a plethora of research fields and industrial applications, these techniques can substantially improve decision making, minimize risk and unearth valuable insights that would otherwise remain hidden. The challenge of effectiveness in uncertain data is to correctly determine the set of possible results, each associated with the correct probability of being a result, in order to give a user a confidence about the returned results. The contrary challenge of efficiency, is to compute these result and corresponding probabilities in an efficient manner, allowing for reasonable querying and mining times, even for large uncertain databases.

The paradigm used to master both challenges, is to identify a small set of equivalent classes of possible worlds, such that members of the same class can be treated as equivalent in the context of a given query predicate or data mining task. In the scope of this work, this paradigm will be formally defined, and applied to the most prominent classes of spatial queries on uncertain data, including range queries, k-nearest neighbor queries, ranking queries and reverse k-nearest neighbor queries. For this purpose, new spatial and probabilistic pruning approaches are developed to further speed up query processing. Furthermore, the proposed paradigm allows to develop the first efficient solution for the problem of frequent co-location mining on uncertain data.

Special emphasis is taken on the temporal aspect of applications using modern data collection technologies. While the aforementioned techniques work well for single points of time, the prediction of query results over time remains a challenge. This thesis fills this gap by modeling an uncertain spatio-temporal object as a stochastic process, and by applying the above paradigm to efficiently query, index and mine historical spatio-temporal data.

Zusammenfassung (Abstract in German)

Moderne Technologien, z.B. Sattelitentechnologie und Technologie in Smart Phones, erzeugen eine Flut räumlicher Geo-Daten. Zudem ist in der Gesellschaft ein Trend zu beobachten diese erzeugten Daten freiwillig auf öffentlich zugänglichen Plattformen zur Verfügung zu stellen. Diese Datenflut hat immenses Potential, um neues und nützliches Wissen zu entdecken. Diese Daten sind jedoch grundsätzlich unsichere räumliche Daten. Die Unsicherheit ergibt sich aus mehreren Aspekten. Zum einen kommt es bei Messungen grundsätzlich zu Messungenauigkeiten, zum anderen ist zwischen diskreten Messzeitpunkten eine Interpolation nötig, die zusätzliche Unsicherheit erzeugt. Auerdem werden die Daten oft absichtlich reduziert, um Speicherplatz und Transfervolumen einzusparen, wodurch weitere Information verloren geht. Diese Unsicherheit schafft einen sofortigen Bedarf für skalierbare und flexible Methoden zur Verwaltung und Auswertung solcher Daten.

Im Rahmen dieser Arbeit sollen effektive und effiziente Techniken zur Ähnlichkeitssuche und zum Data Mining bei unsicheren räumlichen und unsicheren räumlich-zeitlichen Daten erarbeitet werden. Diese Techniken liefern wertvolles Wissen, das auf verschiedenen Forschungsgebieten, als auch bei industriellen Anwendungen zur Entscheidungsfindung genutzt werden kann. Bei der Entwicklung dieser Techniken gibt es zwei Herausforderungen. Einerseits müssen die entwickelten Techniken effektiv sein, um korrekte Ergebnisse und Wahrscheinlichkeiten dieser Ergebnisse zurückzugeben. Andererseits müssen die entwickelten Techniken effizient sein, um auch in sehr groen Datenbanken Ergebnisse in annehmbarer Zeit zu liefern.

Die Dissertation stellt ein neues Paradigma vor, das beide Herausforderungen meistert. Dieses Paradigma identifiziert mögliche Datenbankwelten, die bezüglich des gegebenen Anfrageprädikats äquivalent sind. Es wird formal definiert und auf die relevantesten räumlichen Anfragetypen angewendet, um effiziente Lösungen zu entwickeln. Dazu gehören Bereichanfragen, k-Nächste-Nachbaranfragen, Rankinganfragen und Reverse k-Nächste-Nachbarnanfragen. Räumliche und probabilistische Pruningkriterien werden entwickelt, um insignifikante Ergebnisse früh auszuschlieen. Zudem wird die erste effiziente Lösung für das Problem des "Spatial Co-location Minings" auf unsicheren Daten präsentiert.

Ein besonderer Schwerpunkt dieser Arbeit liegt auf dem temporalen Aspekt moderner Geo-Daten. Während obig genannte Techniken dieser Arbeit für einzelne Zeitpunkt sehr gut funktionieren, ist die effektive und effiziente Verwaltung von unsicheren räumlich-

zeitlichen Daten immer noch ein weitestgehend ungelöstes Problem. Diese Dissertation löst dieses Problem, indem unsichere räumlich-zeitliche Daten durch stochastische Prozesse modeliert werden. Auf diese stochastischen Prozesse lässt sich das oben genannte Paradigma anwenden, um unsichere räumlich-zeitliche Daten effizient anzufragen, zu indexieren, und zu minen.

Part I

Introduction

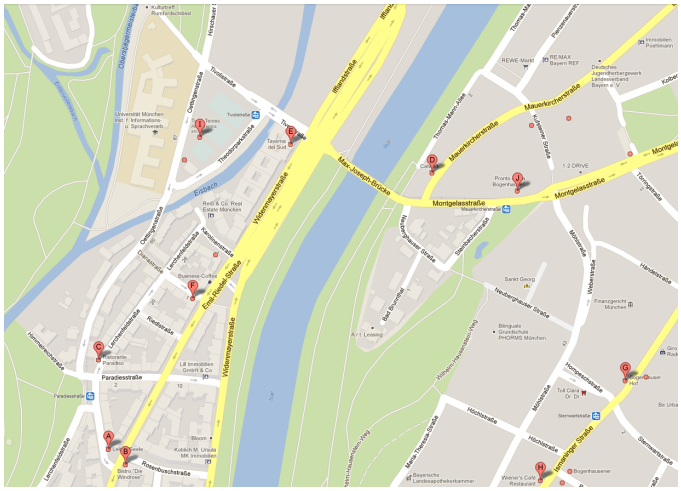


Figure 1: Google Maps: Restaurants near The Chinese Tower, Munich.

Spatial Data

A spatial database system can be defined as a database system that offers spatial objects in its data model and query language, and supports spatial objects in its implementation, providing at least spatial indexing and spatial join methods [79]. A spatial database is optimized to store, query and mine data that is related to objects in space. Figure 1 shows a map from Google Maps (http://maps.google.com) obtained by entering the query "Find restaurants near the Chinese Tower in Munich". It shows different representations of spatial objects such as points, lines and regions. A point may represent a data object for which only its location is important and its extent in space is not important. For example, the balloons labelled A to I point to the locations of restaurants. Lines and polylines represent connections in space (i.e., roads, highways, rivers). Polygons describe spatial regions, such as parks and facilities. In Figure 1, the Institute of Informatics of Ludwig-Maximilians-University and the English Garden are represented by regions. In addition to spatial attributes describing the location of a spatial object, a spatial object generally includes further non-spatial information. Such *geo-enriched data* may include additional information, such as

- numerical attributes, describing for example the age of an object, or the average rent of an apartment building or a region.
- textual attributes, such as names or textual descriptions of the object.

Figure 2: GPS trajectory of an anonymous individual.

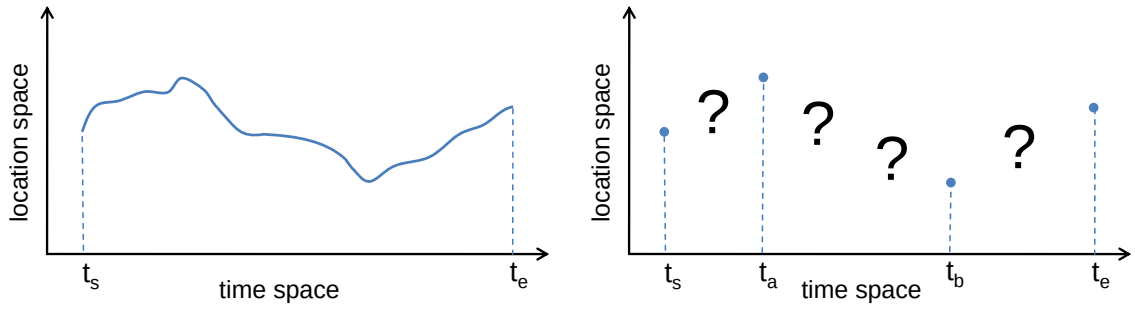
- social information, describing relationships between users and spatial objects. In such a *geo-social network* like facebook[1], Google Latitude [2] and foursquare[3] users can rate spatial objects and recommend them to their friends.
- image and video information of an object, e.g. showing a restaurants from outside and from inside.

The main challenge of querying and mining spatial data, is to combine both spatial and non-spatial attributes. For example, a user u may initiate a query such as *"return all of the user's friends within 100m distance of the user."*. This query returns all objects o that satisfy both the spatial query predicate (o is within 100m range of u) and the non-spatial query predicate (o is a friend of u). An example of a spatial data mining task is to find areas of a city having a low average rent level.

Spatio-Temporal Data

A traditional spatial database does not offer support for objects that change their location over time. However, efficient management of large collections of (location, time) data pertaining to mobile entities whose whereabout change over time is a paramount in a plethora of application domains: from geo-social network applications, through structural and environmental monitoring, disaster/rescue management and remediation, to Geographic Information Systems (GIS) and Tourist Information-Providing (TIP) systems. Database systems are required to capture the time varying nature of the modelled phenomena, and spatial databases must capture the movement of spatial objects over time.

[1] http://www.facebook.com
[2] http://latitude.google.com/
[3] http://foursquare.com



(a) No Uncertainty (b) Uncertainty between discrete observations

Figure 3: Trajectories in space and time

A spatio-temporal database system is database that supports management and analysis of large collections of (location, time) data pertaining to mobile entities whose whereabout changes over time. Figure 2 shows a map from Google Maps enriched with information about the movement data of an anonymous individual over time, which results from check-in data taken from FourSquare (http://foursquare.com). A database that manages such data relating to both space and time information is a *spatio-temporal* database. A common example is a database tracking moving objects, which typically can occupy only a single position at a given time. As an example, a mobile phone that moves among the various cells of the wireless network leaves, during its interactions with the network, a set of triples $(id, loc, time)$, each specifying the localization at space loc and at time $time$ of the phone id. This work uses the common model ([207, 81]) in which a spatio-temporal database is a collection of $(id, loc, time)$ triples, so-called observations, where time is a point in time at which a database object id is known to be at a spatial location location. Starting from the set of triples for a given object id is therefore possible, in principle, to approximate a function

$$id : time \rightarrow space,$$

which assigns a location to object id for each moment in a given time interval. We call such a function a trajectory, as depicted in Figure 3(a) for a one-dimensional space.

The tremendous wealth of information hidden in spatial and spatio-temporal data is emphasized by the recent McKinsey report "Big data: The next frontier for innovation, competition, and productivity" (June 2011) estimating "600 billion USD potential annual consumer surplus from using personal location data globally" [137], thus identifying a great opportunity for industry. Furthermore, our ability to reveal valuable information from spatial and spatio-temporal data will enable scientists of any discipline to gain a new and so far unknown level of knowledge from their data, promoting novel scientific workflows and groundbreaking insights. This will leverage the paradigm of data-driven science (a.k.a. eScience), envisioned by the late Turing Award winner Jim Gray in 2008 as the 4th paradigm of science [87].

Figure 4: Observed past locations of a vehicle and possible future locations.

Uncertainty in Spatial and Spatio-Temporal Data

The task to unearth this wealth of knowledge is not trivial, with one of the main challenges being the inherent uncertainty of spatial and spatio-temporal data. Typical, the source context of spatial databases consists of heterogeneous sensor deployments, including mobile stations, satellite imagery, citizen supplied (crowd sourced) data, ground and aerial LIDAR, and many more types of sources. In addition to many different types of sensors, the same type of sensor is often used redundantly, to measure the same variable from different positions and angles. It is clear that different sensors may yield inconsistent and contradictory information. Traditional database approaches ([13]) to repair such inconsistencies cannot be applied here: Due to the uncertainty, the part of the real world modelled by the database can no longer be expressed by one single version that is guaranteed to be correct. Rather, there exist many possible worlds, each associated with a probability of being correct.

In spatio-temporal databases, we need to consider the case where information about an object can only be measured sporadically, such as in applications where positions of objects are tracked by GPS or RFID technology. Between such observations, the position of the object is not explicitly stored in the database, as depicted in Figure 3(b). Some kind of model is required to calculate and objects current position by using past observations. Such an interpolation, which is called *dead reckoning* in navigation, may give the best available information on the objects position, but is subject to significant errors due to many factors as both speed and direction deteriorate unexpectedly. Furthermore, each estimate of position is relative to the previous one, causing cumulative errors.

Traditional spatial and spatio-temporal database systems simply ignore these aspects of uncertainty, by expressing the database by a single world aggregated for example by using expected values, maximum likelihood or dead reckoning. Yet, this single world may be entirely impossible or it may have a very small probability of being be correct. Clearly, query processing and data mining tasks based on such an aggregation of uncertainty, may

yield misleading and wrong results. To illustrate the problem of aggregated uncertainty, consider the situation illustrated in Figure 4, showing a vehicle that has been observed northwest of lake "Emmeringer See" at time 12:05 and at time 12:06 at the depicted positions. At a later time 12:07, no further observation of the vehicle has been made, such that its location at time 12:07 is uncertain. A simple dead reckoning approach using linear interpolation assumes that the object continues to drive in the very same direction it has taken between time 12:05 and 12:06, causing the vehicle to be located in the lake at time 12:07. Accounting for the fact that the vehicle may have taken either a left turn, or a right turn at the depicted road intersection, there will be two depicted possible positions of the vehicle. Without any further knowledge about the likelihood of a left (right) turn, we must assume a uniform distribution, leading to a 0.5 probability of the vehicle to be at either position. A simple aggregation, that reduces this random position to an expected position, would yield a useless result, putting the vehicle into the lake once more.

This shortcoming of traditional spatial and spatio-temporal database systems raises an urgent need for approaches to directly utilize uncertainty information, by considering all possible worlds. Such probabilistic approaches have the potential to significantly improve the quality of spatial database systems, by allowing to model knowledge about the real world that is omitted in traditional spatial database systems. The challenge of this thesis is to identify solutions to reap the potential benefits of this probabilistic information to more effectively query and mine spatial and spatio-temporal data. Solving this challenge will be a major milestone leading to the greater vision of combining spatial and non-spatial data into geo-enriched data to create new application to enhance everyday's life.

Outline

For this purpose, this thesis is subdivided as follows. The next part, Part II, will give a *survey* on the field of *managing, querying and mining uncertain and spatial data*, explaining the various concepts required to understand this thesis. Furthermore, it will introduce the *paradigm of equivalent worlds*, a general concept that facilitates development of efficient algorithms for problems involving uncertain data. In Part III, the paradigm of equivalent worlds is be applied to find efficient solutions for the most relevant types of *spatial queries on uncertain data*, including range queries, k-nearest neighbor queries, ranking queries and reverse k-nearest neighbor queries. Part IV applies this paradigm to give efficient solutions for the important problem of spatial data mining of *spatial-collocation mining*. In Part V, the temporal aspect of uncertain spatial data is considered, leading to *uncertain spatio-temporal data*. New solutions to efficiently *query, index and mine uncertain spatio-temporal data* are presented, based upon the paradigm of equivalent worlds. Part VI envisions future research directions, by pushing the results of this thesis one step ahead, to efficiently handle uncertainty in semantically enriched geo-spatial data. Part VII concludes this thesis.

Preliminary results of this thesis have been published as full papers at various database and data-mining conferences and journals, including SIGMOD ([65]), VLDB/PVLBD ([23, 141]), SIGKDD ([28]), ICDE ([22, 66]), CIKM ([66]), SDM ([113]) and TKDE ([25]).

Furthermore, the introductional parts of this thesis have been presented to the research community as slides in the scope of tutorials. A tutorial on querying and mining of uncertain spatial data, for which I have prepared a majority of presentation slides, has been presented to a broad audience at VLDB Conference 2010 ([157]), while a tutorial on the management and mining of spatio-temporal data, that I will co-present, has been accepted for presentation in the scope of a tutorial at ICDE Conference 2014. Finally, one of the future visions presented in Section VI is given opportunity to be discussed at a new workshop ([140]) that is held in conjunction with ACM SIGMOD Conference 2014 and for which I am co-chairing the programm.

At the beginning of each of the main parts of this thesis (Parts II - VI), a road-map is given, summarizing the chapters of each part, and indicating whether and where parts of these chapters have been published. This thesis is self-contained, such that no knowledge of the aforementioned papers is required to read this thesis. To achieve this self-containment, introductions and tutorials to the fields of spatial, spatio-temporal and uncertain database management are given in this thesis. Experts may skip the preliminary part of this thesis. Chapters presenting research results unpublished at the time of submission of this thesis are marked as such.

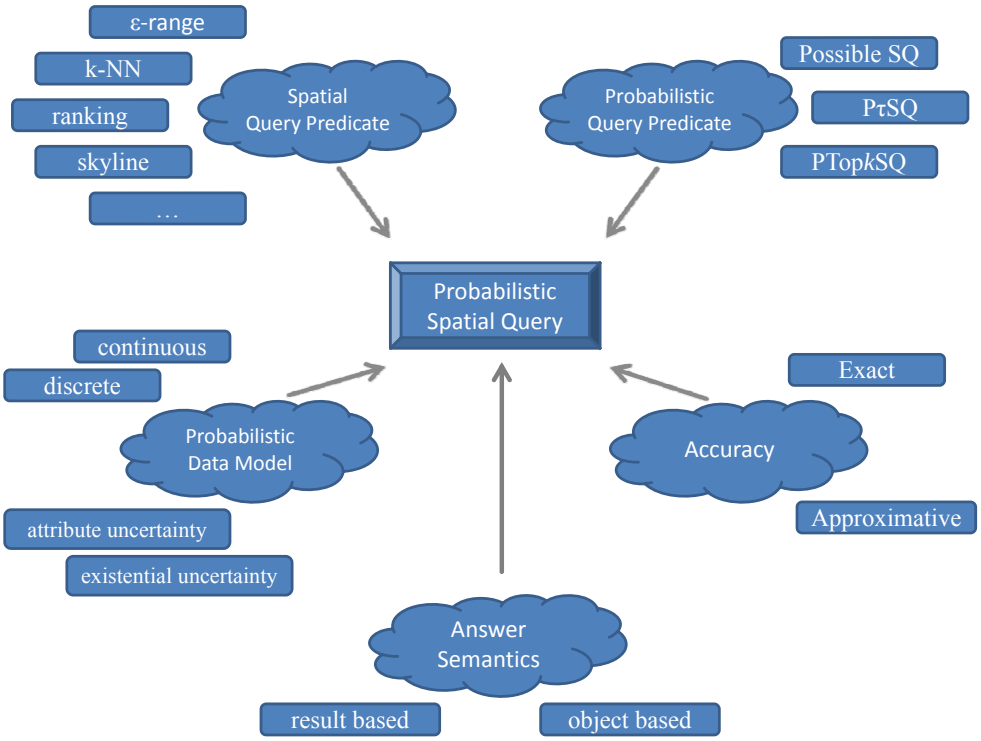
Further publications on the field of querying and mining uncertain data that emerged during the course of this PhD contain extensions that are discussed only briefly ([30, 64, 24]) in this thesis, or are omitted entirely ([26, 71, 29]) for brevity and to control the extents of this document. These extensions are not necessary for the main concepts presented in this thesis.

More publications by the author of this thesis consider spatial data without any notion of uncertainty ([107, 5, 108, 61, 63]), and therefore fail to qualify for the scope of this thesis. Nevertheless, the experience gained while working on the field of spatial data management has highly benefited this thesis.

All these publications have their list of authors sorted in alphabetical order, following the tradition of the group of Prof. Dr. Hans-Peter Kriegel.

Part II

Spatial and Uncertain Data: Preliminaries



This part of this thesis gives a survey on the field of modeling, managing and querying uncertain spatial data. It is subdivided into the following chapters:

- Chapter 1 formally defines *spatial data* and introduces the most relevant *spatial similarity query predicates* including *spatial ϵ-range queries*, *k-nearest neighbor queries*, and *reverse k-nearest neighbor queries*. For each spatial similarity query type, efficient solutions in the presence of uncertainty are present later, in Chapters 4-8.

- Before these solutions can be presented, the concepts of managing, modeling and querying uncertain data are elaborated in Chapter 2. Parts of this chapter have been presented in the form of presentation slides on our conference tutorial held in 2010 at the 36th International Conference on Very Large Data Bases ([157]). This chapter is subdivided into a number of sections in order to give a survey of definitions, notions and techniques used in the field of querying and mining uncertain spatio-temporal data. Section 2.1 presents a survey of state-of-the-art *data representations models* used in the field of uncertain data management. To answer any query on uncertain data, well-defined semantics of such queries are required. Therefore, Section 2.3 introduces the *possible world semantics* for uncertain data, widely used in related work as well as in the remainder of this thesis. Given an uncertain database, the result of a probabilistic query can be interpreted in two ways as elaborated in Section 2.4. This distinction between different *probabilistic answer semantics* is not made explicitly in any related work, but is required to gain a deep understanding of problems in the field of querying uncertain spatial data and their complexity. Furthermore, a probabilistic query is required to specify a *probabilistic query predicate*. A probabilistic query predicate defines the requirements for a candidate result to be sufficiently stochastic significant to be returned as a query result. Probabilistic query predicates described in Section 2.5 include possibilistic queries, probabilistic threshold queries and probabilistic top-k queries. Section 2.6 explains Monte-Carlo sampling based probability approximation techniques.

- Chapter 3 introduces a novel paradigm for uncertain data to efficiently answer any kind of query using possible world semantics. This *Paradigm of Equivalent Worlds* generalizes existing solutions by identifying requirements a query must satisfy in order to have a polynomial solution. For any query satisfying these requirements, a general framework is presented to find an efficient solution. All solutions given in the remainder of this thesis will be based on this general paradigm. This general paradigm of approaching uncertain data is first published in this thesis. Following the paradigm of equivalent worlds, Section 3.3 presents efficient solutions for the problem of computing the sum of a Poisson-binomial distributed random variable. This Section presents existing techniques to solve this problem and explicitly shows how these existing solutions implicitly apply the paradigm of equivalent worlds. The techniques shown in this chapter will be paramount to develop novel solutions to efficiently answer similarity queries on uncertain spatial data in Part III.

Chapter 1

Spatial Data

Objects in a spatial database can be points, lines and polygons. This thesis focuses on the case of spatial points, as in most application where data uncertainty is involved, the extent of objects is either non-existent or non-relevant. For instance, GPS signals are represented by longitude and latitude values, i.e., a point in the two-dimensional geo-space. Analogously, traffic management system tracking the position of vehicles using RFID technology, can only measure the location of an object, but not its extent. Furthermore, spatial objects for which their spatial extent is relevant, such as building and lakes, do not change their position frequently, such that their is usually little uncertainty involved in the management of objects represented by lines and polygons.

The assumption that objects are represented by points leads to the following simple definition of a spatial database.

Definition 1. *A spatial database* $\mathcal{DB} = \{o_1, ..., o_N\}$ *consists of a set of* $N := |\mathcal{DB}|$ *spatial objects. Each spatial object* $o_i \in \mathbb{R}^d$ *is represented by a d-dimensional real vector corresponding to the location of* o_i *in a d-dimensional space.*

A main requirement for a spatial database management system is to provide efficient support for the tasks of spatial similarity search. In a nutshell, the task of *spatial similarity search* is to find all objects in the database similar to a given query object. This chapter recapitulates the various types of *similarity* queries commonly used in spatial databases.

1.1 Spatial Similarity Queries

A spatial similarity query returns, for a given spatial database $\mathcal{DB}$ and a spatial query object q, the set of all objects in $\mathcal{DB}$ which are similar to q. However, the semantic of similarity between two objects may differ between applications. For example, consider the following three spatial applications:

I In a geo-social network, a user wishes to find the set of friends, who are in close spatial vicinity (i.e., spatially similar to) to his favorite pub.

II In a road network, the driver of a vehicle wishes to find the gas station closest to him.

III To open a pizza restaurant, the owner wishes to find the location in a city where he influences the largest number of customers.

In Application I, the aim is to find all friends that are close enough, thus having a spatial distance of less than some given threshold ϵ.[1] The corresponding query type is an ϵ-range query. In contrast, an *epsilon*-range query may be inappropriate for Application II. The set of all gas station in range of the limit of the vehicles current fuel level may be very large. Picking one result at random may result in a gas station that is barely in range and thus, causing the user to run out of fuel if this gas station happens to be closed. Rather, Application II requires to return one (or more) gas station(s) in $\mathcal{DB}$ having the smallest distance to the vehicle. This type of query is denoted as a k-nearest neighbor (kNN) query. In Application I, a k-nearest neighbor query may not make much sense, as the k friends of yours having the smallest distance to you may still be too far away if there is less than k of your friends close to your pub. Or, there may be more than k of your friends close by, resulting in some of them not being returned as a result. Finally, Application III requires, for a set of alternative locations for the new shop, to find the location for which it holds that a large number of potential customers (i.e., a large number of apartment buildings) have the new pizza restaurant as one of their closest restaurants. This type of query is denoted as a reverse k-nearest neighbor (RkNN) query. Using a k-nearest neighbor query in Application III will yield a result of k-nearest restaurants for each possible shop location, thus not giving any new insights. However, a RkNN query may return more (or less) results than k, giving an indication of how many database objects are influenced by the query object q.

In the following, the three presented query types will be formally defined. Therefore, let $dist(\cdot, \cdot)$ be a spatial distance metric, such as Euclidean distance, distance on a road network, fuel consumption on a road network, etc.

[1] In such an application, the social distance may also be considered, such that close friends are allowed to have a larger spatial distance in order to still be returned as a result. This interesting aspect of geo-social data is omitted here for brevity.

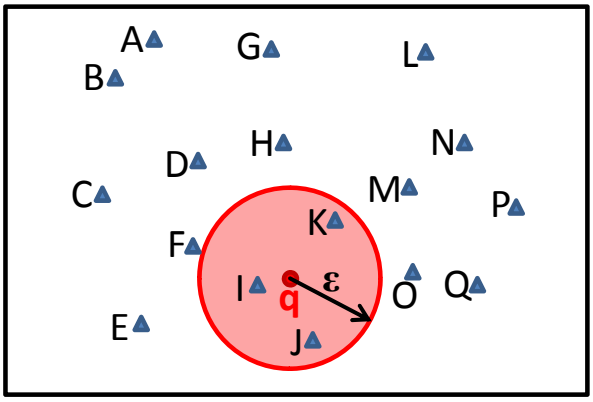


Figure 1.1: A spatial ϵ-range query.

1.1.1 The Spatial Range Query

The most commonly used query type in spatial databases is the spatial range query. In this work, we consider the most prominent types of spatial range queries, namely the ϵ-range query and the window query.

Definition 2 (ϵ-Range Query)**.** *Let $\mathcal{DB}$ be a spatial database, let q be a query point, and let ϵ be a positive real value. A ϵ-range query returns all objects in $\mathcal{DB}$ having a distance of at most ϵ to q, i.e.,*

$$\epsilon\text{-}range(q) = \{o \in \mathcal{DB} | dist(q, o) \leq \epsilon\},$$

Example 1. *In Figure 1.1 a database of spatial point objects and a spatial query point q are depicted. The circle around q with radius ϵ highlights the space having a Euclidean distance of at most ϵ to q. The set $\{I, J, K\}$ of objects containing only the database objects located in this circle are returned by this ϵ-range query.*

A query type closely related to the ϵ-range query is the window query.

Definition 3. *[Spatial Window Query] Let $\mathcal{DB}$ be a spatial database, let q be a query point, and let δ_X and δ_Y be positive real values. A spatial window query returns all objects located in the region having a vertical distance of at most δ_X, and a horizontal distance of at most δ_Y.*

$$\square(q, \delta_X, \delta_Y) = \{o \in \mathcal{DB} | abs(q.X - o.X) \leq \delta_X \wedge abs(q.Y - o.Y) \leq \delta_Y\}.$$

Thus, the main difference between the ϵ-range query and the spatial window query, is the shape of the query region. Note that in the literature, the query rectangle is often given by two points (X_{min}, Y_{min}) and (X_{max}, Y_{max}), which is an equivalent representation of a rectangle. This representation can be transformed into the representation of Definition 3 by defining $q := (\frac{X_{min}+X_{max}}{2}, \frac{Y_{min}+Y_{max}}{2})$, $\delta_X = \frac{X_{min}-X_{max}}{2}$ and $\delta_Y = \frac{Y_{min}-Y_{max}}{2}$.

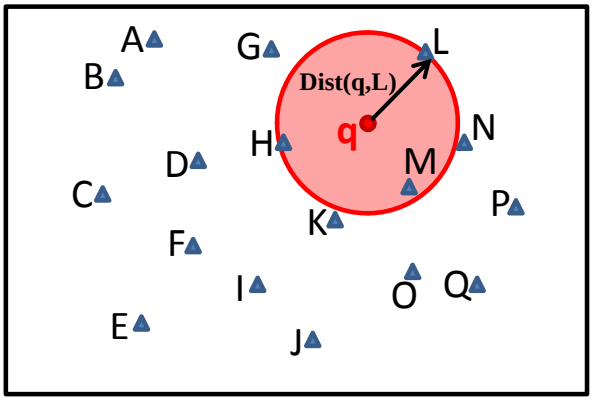


Figure 1.2: A spatial 3-nearest neighbor query.

1.1.2 The k-Nearest Neighbor Query

A spatial ϵ-range query require the user to specify a proper value of ϵ, in order to get a proper number of relevant results: a value of ϵ that is chosen too small may yield few or no results at all, while a value of ϵ chosen too large may return too many results or even the whole database. To guarantee a certain result size, a k-Nearest Neighbor (kNN) query is defined as follows.

Definition 4 (k-Nearest Neighbor Query)**.** *Let $\mathcal{DB}$ be a spatial database, let q be a query point, and let k be a positive integer. A k-nearest neighbor query returns the smallest set $kNN(q, \mathcal{DB})$ of at least k objects in $\mathcal{DB}$ such that*

$$\forall o \in \mathcal{DB} \setminus kNN(q), \forall p \in kNN(q) : dist(q, p) \leq dist(q, o).$$

The distance

$$kNN\text{-}dist(q) := max_{o \in kNN(q)} dist(q, o)$$

is called the kNN-distance of q.

Example 2. *In Figure 1.2, the 3-nearest neighbor set of query object q is $3NN(q, \mathcal{DB}) = \{H, L, M\}$, since for any object $X \in \{H, L, M\}$ and for any object $Y \in \mathcal{DB} \setminus \{H, L, M\}$ it holds that $dist(q, X) \leq dist(q, Y)$. The depicted circle illustrates the space having a distance of less or equal to the $3NN$ distance of q. The $3NN(q, \mathcal{DB})$ query contains all of the objects and only the objects in this circle.*

Unlike for a ϵ-range query, the range in which results of a kNN query can be found depends on the query object q, as different query objects may have a different distance to their k'th nearest neighbor. Note that in the definition above, the number of results of a kNN query may be greater than k in the case of ties, i.e., in the case where multiple objects have a distance to q identical to the kNN distance of q. In applications that require to return exactly k results, ties can be broken arbitrarily, by iteratively dropping objects

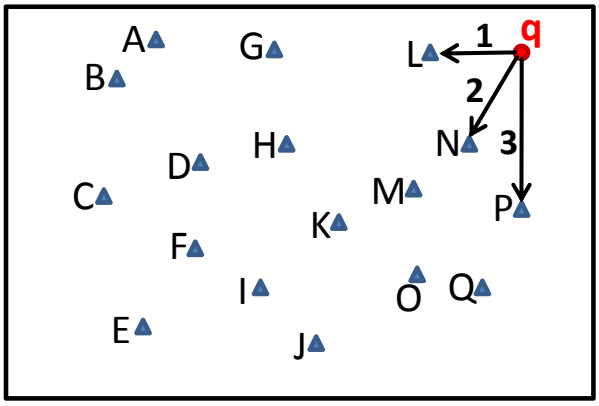


Figure 1.3: A spatial ranking query.

having a distance to q identical to the kNN distance of q, until k objects are left. In many applications, the task may to return the k-nearest neighbors sorted increasingly by their distance to q. This task defines similarity ranking queries as follows.

Definition 5 (Similarity Ranking Query)**.** *Let $\mathcal{DB}$ be a spatial database, let q be a query point, and let k be a positive integer. A similarity ranking query returns a list*

$$rank(q, k, \mathcal{DB}) = [o_{p_1} \in kNN(q, \mathcal{DB}), ..., o_{p_k} \in kNN(q, \mathcal{DB})]$$

of length k sorted in increases order of their distance to q, such that

$$\forall o \in \mathcal{DB} \setminus rank(q, k, \mathcal{DB}), \forall p \in rank(q, k, \mathcal{DB}) : dist(q, p) \leq dist(q, o),$$

and

$$\forall 1 \leq i \neq j \leq k : o_{p_i} \neq o_{p_j} \Rightarrow dist(q, o_{p_i}) \leq dist(q, o_{p_i}).$$

The main challenge of similarity ranking queries is that in most applications, k is not known at query time. Therefore, a ranking query iteratively returns individual results to the user, until the user signals that no more results are needed. This kind of query is often referred to *incremental similarity ranking query*. The algorithmic challenge of such queries is to avoid running a complete k-nearest neighbor query from scratch in each iteration. Rather, efficient solutions are required to incrementally determine the result of $rank(q, k, \mathcal{DB})$ given the result of $rank(q, k-1, \mathcal{DB})$ as well as intermediate results acquired during previous iterations.

Example 3. *Figure 1.3 shows the result of a ranking query for the depicted query object q. Object L is the first object that is returned, being the nearest neighbor of q. If requested by the user, object N is returned second, and object P is returned third.*

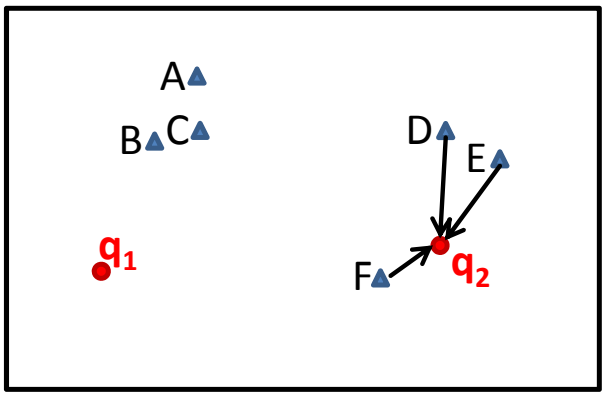


Figure 1.4: A 3-reverse nearest neighbor query.

1.1.3 The Reverse k-Nearest Neighbor Query

Definition 6 (Reverse k-Nearest Neighbor Query)**.** *Let $\mathcal{DB}$ be a spatial database, let q be a query point, and let k be a positive integer. A reverse k-nearest neighbor query returns the set of of objects having q as one of their k-nearest neighbor, i.e.,*

$$RkNN(q) = \{o \in \mathcal{DB} \setminus \{q\} | q \in kNN(o)\} \text{ if } q \in \mathcal{DB} \text{ and}$$

$$RkNN(q) = \{o \in \mathcal{DB} | dist(q,o) \leq kNN\text{-}dist(q)\} \text{ if } q \notin \mathcal{DB}.$$

Example 4. *For $k = 2$, object q_1 in Figure 1.4 has no reverse nearest neighbor, since no database object has q_1 as one of its two nearest neighbors. In particular, the objects A, B and C that are nearest to q_1, have each other as their two-nearest neighbor sets. In contrast, object q_2 has objects D, E and F as two-reverse nearest neighbors, since all of these objects contain q_2 in their two-nearest neighbor set.*

It is important to note that the kNN relation $kNN_{\mathcal{DB}} := \{(r \in \mathcal{DB}, s \in DB \setminus \{r\}) | s \in kNN(r)\}$ is not identical to the RkNN relation $RkNN_{\mathcal{DB}} := \{(r \in \mathcal{DB}, s \in \mathcal{DB} \setminus \{r\}) | s \in RkNN(r)\}$. This observation follows direct from the fact that the relation $kNN_{\mathcal{DB}}$ is not symmetrical, i.e., $(o_i, o_j \in kNN_{\mathcal{DB}})$ does not imply $(o_j, o_i \in kNN_{\mathcal{DB}})$.

Example 5. *For example, in Figure 1.4, object q_1 has object B as its 1-nearest neighbor. However, object B does not have object q_1 as its 1-nearest neighbor since objects A and C are closer to B than q_1.*

A further type of spatial similarity query is the spatial skyline query[37], which is not featured in this dissertation.

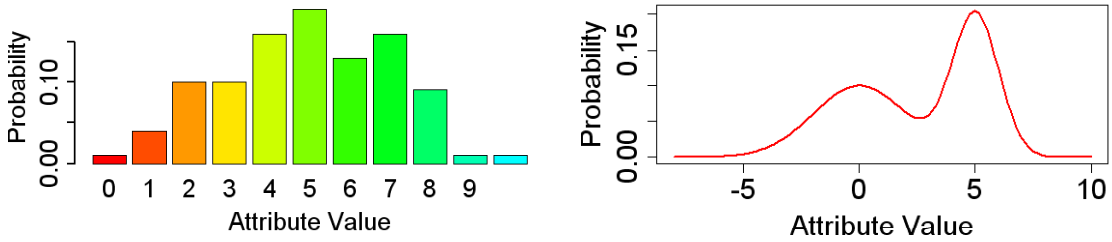
Chapter 2

Uncertain Data

2.1 Discrete and Continuous Models for Uncertain Data

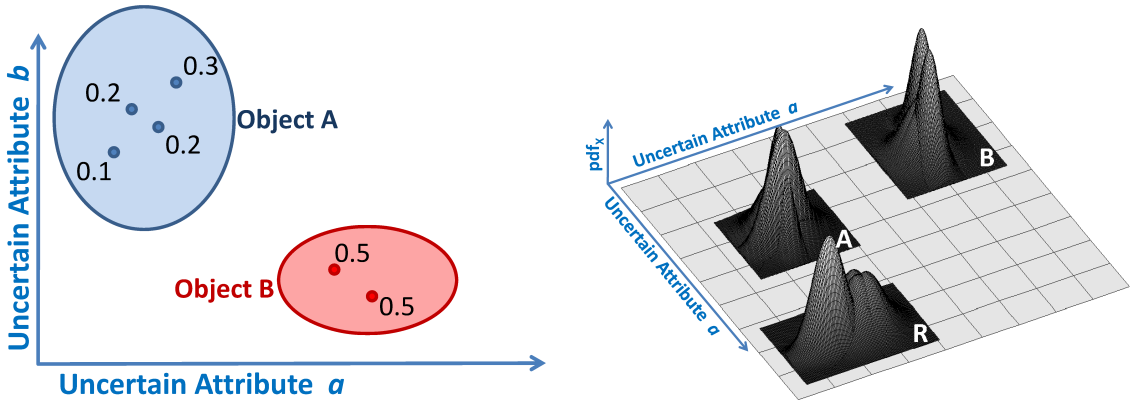
An object is uncertain if at least one attribute of o is uncertain. The uncertainty of an attribute can be captured in a discrete or continuous way. A discrete model uses a probability mass function (pmf) to describe the location of an uncertain object. In essence, such a model describes an uncertain object by a finite number of alternative instances, each with an associated probability [110, 147], as shown in Figure 2.1(a). In contrast, a continuous model uses a continuous probability density function (pdf), like Gaussian, uniform, Zipfian, or a mixture model, as depicted in Figure 2.1(b), to represent object locations over the space. Thus, in a continuous model, the number of possible attribute values is uncountably infinite. In order to estimate the probability that an uncertain attribute value is within an interval, integration of its pdf over this interval is required [177]. The random variables corresponding to each uncertain attribute of an object o can be arbitrarily correlated.

To capture positional uncertainty, such models can be applied by treating longitude and latitude (and optionally elevation) as two (three) uncertain attributes. In the case of discrete positional uncertainty, the position of an object A is given by a discrete set



(a) Discrete Probability Mass Function (b) Continuous Probability Density Function

Figure 2.1: Models for Uncertain Attributes



(a) Discrete Case (b) Continuous Case

Figure 2.2: Uncertain Objects

$a_1, ..., a_m$ of $m \in \mathbb{N}$ possible alternatives in space, as exemplarily depicted in Figure 2.2(a) for two uncertain objects A and B. Each alternative a_i is associated with a probability value $p(a_i)$, which may for example be derived from empirical information about the turn probabilities of intersection in an underlying road network. In a nutshell, the position A is a random variable, defined by a probability mass function pdf_A that maps each alternative position a_i to its corresponding probability $p(a_i)$, and that maps all other positions in space to a zero probability. An important property of uncertain spatial databases is the inherent correlation of spatial attributes. In the example shown in Figure 2.2(a) it can be observed that the uncertain attributes a and b are highly correlated: given the value of one attribute, the other attribute is certain, as there is no two alternatives of objects A and B having identical attribute values in either attribute.

Clearly, it must hold that the sum of probabilities of all alternatives must sum to at most one:

$$\sum_{i=1}^{m} p(a_i) \leq 1$$

In the case where $\sum_{i=1}^{m} p(a_i) \leq 1$ object A has a non-zero probability of $1 - \sum_{i=1}^{m} p(a_i) \geq 0$ to not exist at all. This case is called *existential uncertainty*, and A is denoted as *existentially uncertain* [205]. If the total number of possible instances m is greater than one, A is denoted as *attribute uncertain*. In the context of uncertain spatial data, attribute uncertainty is also referred to as *positional uncertainty* or *location uncertainty*. An object can be both existentially uncertain and attribute uncertain. In Figure 2.2(a), object A is both existentially uncertain and attribute uncertain, while object B is attribute uncertain but does exist for certain.

In the case of continuous uncertainty, the number of possible alternative positions of an object A is infinite, and given by the non-zero domain of the probability density function

pdf_x. The probability of A to occur in some spatial region r is given by integration

$$\int_r pdf_A(x)dx.$$

Since arbitrary pdfs may be represented by an infinitely large number of $(position, probability)$ pairs, such pdfs may require infinite space to represent. For this reason, assumptions on the shape of a pdf are made in practice. All continuous models for positionally uncertain data therefore use parametric pdfs, such as Gaussian, uniform, Zipfian, mixture models, or parametric spline representations. For illustration purpose, Figure 2.2(b) depicts three uncertain objects modelled by a mixture of gaussian pdfs. Similar to the discrete case, the constraint

$$\int_{\mathbb{R}^d} pdf_A(x)dx \leq 1$$

must be satisfied, where $\mathbb{R}^d$ is a d dimensional vector space. In the case of spatial data, d usually equals two or three. The notion of existentially and attribute uncertain objects is defined analogous to the discrete case.

The following section reviews related work and state-of-the-art on the field of modeling uncertain data.

2.2 Existing Models for Uncertain Data

This section gives a brief survey on existing models for uncertain spatial data used in the database community. Many of the presented models have been developed to model uncertainty in relational data, but can be easily adapted to model uncertain spatial data. Since one of the main challenges of modeling uncertain data is to capture correlation between uncertain objects, this section will elaborate details on how state-of-the-art approaches tackles this challenge. Both discrete and continuous models are presented.

Discrete Models

In addition to reviewing related work defining discrete uncertainty models, the aim of this section is to put these papers into context of Section 2.1. In particular, models which are special cases or equivalent to the model presented in Section 2.1 will be identified, and proper mappings to Section 2.1 will be given.

Independent Tuple Model. Initial models have been proposed simultaneously and independently in [74, 217]. These works assume a relational model in which each tuple is associated with a probability describing its existential uncertainty. All tuples are considered independent from each other. This simple model can be seen as a special case of the model presented in Section 2.1, where only existential uncertain but no attribute uncertainty is modelled.

Block-Independent Disjoint Tuples Model and X-Tuple model A more recent and the currently most prominent approach to model discrete uncertainty is the

block-independent disjoint tuples model ([56]), which can capture mutual exclusion between tuples in uncertain relational databases. A probabilistic database is called block independent-disjoint if the set of all possible tuples can be partitioned into blocks such that tuples from the same block are disjoint events, and tuples from distinct blocks are independent. A commonly used example of a block-independent disjoint tuples model is the *Uncertainty-Lineage Database Model*([16, 163, 172, 202, 203]), also called *X-Relation Model* or simply *X-Tuple Model* that has been developed for relational data. In this model, a probabilistic database is a finite set of *probabilistic tables*. A probabilistic table T contains a set of (uncertain) tuples, where each tuple $t \in T$ is associated with a membership probability value $Pr(t) > 0$. A *generation rule* R on a table T specifies a set of mutually exclusive tuples in the form of $R : t_{r_1} \oplus ... \oplus t_{r_m}$ where $t_{r_i} \in T(1 \leq i \leq m)$ and $P(R) := \sum_{i=1}^{m} t_{r_i} \leq 1$. The rule R constrains that, among all tuples $t_{r_1}, ..., t_{r_m}$ involved in the rule, at most one tuple can appear in a possible world. The case where $P(R) < 1$ the probability $1 - P(R)$ corresponds to the probability that no tuple contained in rule R exists. It is assumed that for any two rules R_1 and R_2 it holds that R_1 and R_2 do not share any common tuples, i.e., $R_1 \cap R_2 = \emptyset$. In this model, a possible world w is a subset of T such that for each generation rule R, w contains exactly one tuple involved in R if $P(R) = 1$, or w contains 0 or 1 tuple involved in R if $Pr(R) < 1$.

This model can be translated to a discrete model for uncertain spatial data as discussed in Section 2.1 by interpreting the set T as the set of all possible locations of all objects, and interpreting each rule R as an uncertain spatial object having alternatives t_{r_i}. The constraint that no two rules may share any common tuples translates into the assumption of mutually independent spatial objects. Finally, the case $P(R) < 1$ corresponds to the case of existential uncertainty (see Section 2.1).

A similar block-independent disjoint tuples model is called *p-or-set* [156] and can be translated to the model described in Section 2.1 analogously. In [11], another model for uncertainty in relational databases has been proposed that allows to represent attribute values by sets of possible values instead of single deterministic values. This work extends relational algebra by an operator for computing possible answers. A normalized representation of uncertain attributes, which essentially splits each uncertain attribute into a single relation, a so-called U-relation, allows to efficiently answer projection-selection-join queries. The main drawback of this model is that it is not possible to compute probabilities of the returned possible answers. Sen and Deshpande [166] propose a model based on a probabilistic graphical model, for explicitly modeling correlations among tuples in a probabilistic database. Strategies for executing SQL queries over such data have been developed in this work. The main drawback of using the proposed graphical model is its complexity, which grows exponential in the number of mutually correlated tuples. This is a general drawback for graphical models such as Bayesian networks and graphical Markov models, where even a *factorized representation* may fail to reduce the complexity sufficiently: The idea of a factorized representation is to identify conditional independencies. For example, if a random variable C depends on random variables A and B, then the distribution of C has to be given relative to all combination of realizations of A and B. If however, C is conditionally independent of A, i.e., B depends on A, C depends on B, and C only

transitively depends on A, then it is sufficient to store the distribution of C relative only to the realizations of B. Nevertheless, if for a given graphical model a random variable depends on more than a hand-full of other random variables, then the corresponding model will become infeasible.

And/Xor Tree Model. A very recent work by Li and Deshpande [123] extends the block-independent disjoint tuples model by adding support for mutual co-existence. Two events satisfy the mutual co-existence correlation if in any possible world, either both happen or neither occurs. This work allows both mutual exclusiveness and mutual co-existence to be specified in a hierarchical manner. The resulting tree structure is called an *and/xor tree*. While theoretically highly relevant, the and/xor tree model becomes impracticable in large database having non-trivial object dependencies, as it grows exponentially in the number of database objects.

If not stated otherwise, this thesis will apply the block-independent disjoint tuples model as model of choice for discrete uncertain data.

Continuous Models

In general, similarity search methods based on continuous models involve expensive integrations of the PDFs, hence special approximation and indexing techniques for efficient query processing are typically employed [52, 177]. In order to increase quality of approximations, and in order to reduce the computational complexity, a number of models have been proposed making assumptions on the shape of object PDFs. Such assumptions can often be made in applications where the uncertain values follow a specific parametric distribution, e.g. a uniform distribution [50, 48] or a Gaussian distribution [48, 59, 146]. Multiple such distributions can be mixed to obtain a mixture model [193, 35]. To approximate arbitrary PDFs, [124] proposes to use polynomial spline approximations.

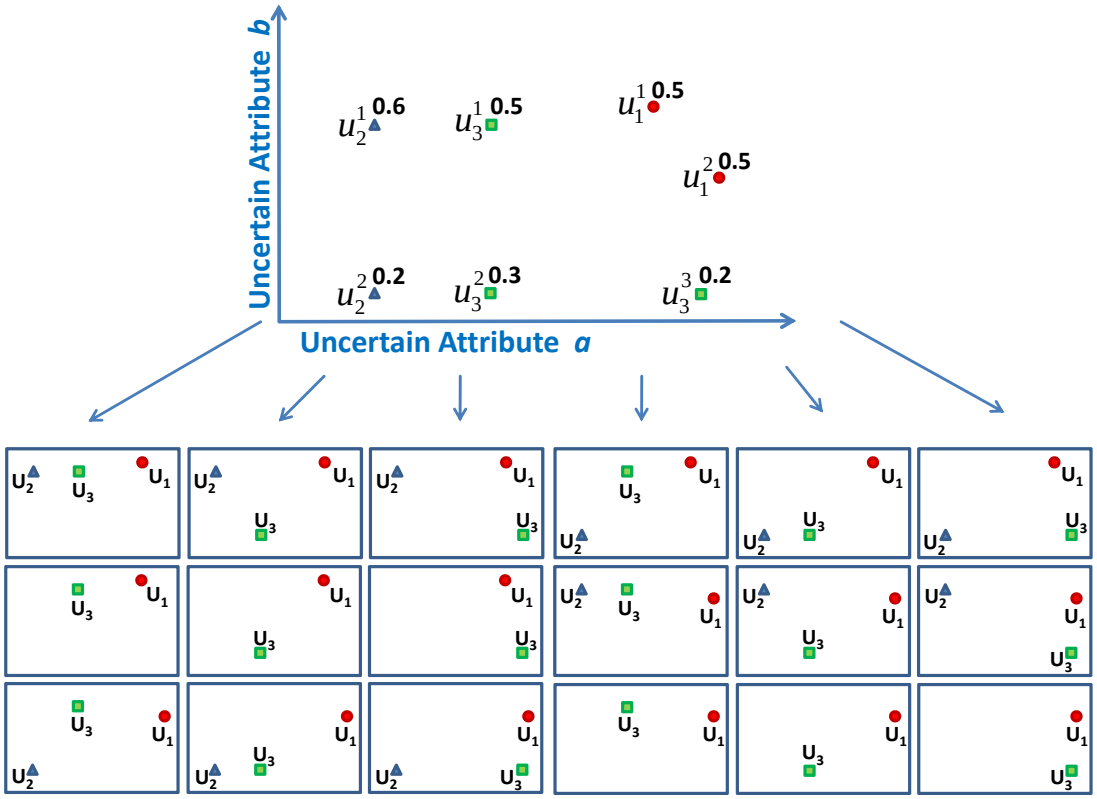


Figure 2.3: An uncertain database and all of its possible worlds.

2.3 Possible World Semantics

In an uncertain spatial database $\mathcal{DB} = \{U_1, ..., U_N\}$, the location of an object is a random variable. Consequently, if there is at least one uncertain object, the data stored in the database becomes a random variable. To interpret, that is, to define the semantics of a database that is, in itself, a random variable, the concept of *possible worlds* is described in this section.

Definition 7 (Possible World Semantics)**.** *A possible world* $w = \{u_1^{a_1}, ..., u_N^{a_N}\}$ *is a set of instances containing at most one instance* $u_i^{a_i} \in U_i$ *from each object* $U_i \in \mathcal{DB}$*. The set of all possible worlds is denoted as* $\mathcal{W}$*. The total probability of an uncertain world* $P(w \in \mathcal{W})$ *is derived from the chain rule of conditional probabilities:*

$$P(w) := P(\bigwedge_{u_i^{a_i} \in w} U_i = u_i^{a_i}) = \prod_{i=1}^{N} P(u_i^{a_i} | \bigwedge_{j<i} u_j^{a_j}). \tag{2.1}$$

By definition, all worlds w *having a zero probability* $P(w) = 0$ *are excluded from the set of possible worlds* $\mathcal{W}$*. Equation 2.1 can be used if conditional probabilities of the position of objects given the position of other objects are known, e.g. by a given graphical model*

Table 2.1: Possible worlds corresponding to Figure 2.3.

World	Probability	World	Probability
$\{u_1^1, u_2^1, u_3^1\}$	$0.5 \cdot 0.7 \cdot 0.5 = 0.175$	$\{u_1^2, u_2^1, u_3^1\}$	$0.5 \cdot 0.7 \cdot 0.5 = 0.175$
$\{u_1^1, u_2^1, u_3^2\}$	$0.5 \cdot 0.7 \cdot 0.3 = 0.105$	$\{u_1^2, u_2^1, u_3^2\}$	$0.5 \cdot 0.7 \cdot 0.3 = 0.105$
$\{u_1^1, u_2^1, u_3^3\}$	$0.5 \cdot 0.7 \cdot 0.2 = 0.07$	$\{u_1^2, u_2^1, u_3^3\}$	$0.5 \cdot 0.7 \cdot 0.2 = 0.07$
$\{u_1^1, u_2^2, u_3^1\}$	$0.5 \cdot 0.2 \cdot 0.5 = 0.05$	$\{u_1^2, u_2^2, u_3^1\}$	$0.5 \cdot 0.2 \cdot 0.5 = 0.05$
$\{u_1^1, u_2^2, u_3^2\}$	$0.5 \cdot 0.2 \cdot 0.3 = 0.03$	$\{u_1^2, u_2^2, u_3^2\}$	$0.5 \cdot 0.2 \cdot 0.3 = 0.03$
$\{u_1^1, u_2^2, u_3^3\}$	$0.5 \cdot 0.2 \cdot 0.2 = 0.02$	$\{u_1^2, u_2^2, u_3^3\}$	$0.5 \cdot 0.2 \cdot 0.2 = 0.02$
$\{u_1^1, u_3^1\}$	$0.5 \cdot 0.1 \cdot 0.5 = 0.025$	$\{u_1^2, u_3^1\}$	$0.5 \cdot 0.1 \cdot 0.5 = 0.025$
$\{u_1^1 u_3^2\}$	$0.5 \cdot 0.1 \cdot 0.3 = 0.015$	$\{u_1^2, u_3^2\}$	$0.5 \cdot 0.1 \cdot 0.3 = 0.015$
$\{u_1^1, u_3^3\}$	$0.5 \cdot 0.1 \cdot 0.2 = 0.01$	$\{u_1^2, u_3^3\}$	$0.5 \cdot 0.1 \cdot 0.2 = 0.01$

such as a Bayesian network or a Markov model. In many applications where independence between object locations can be assumed, as well as in applications where only the marginal probabilities $P(u_i^{a_i})$ are known, and thus independence has to be assumed due to lack of better knowledge of a dependency model, the above equation simplifies to

$$P(w) = \prod_{i=1}^{N} P(u_i^{a_i}). \tag{2.2}$$

Example 6. *As an example, consider Figure 2.3 where a database consisting of three uncertain objects $\mathcal{DB} = \{U_1, U_2, U_3\}$ is depicted. Objects $U_1 = \{u_1^1, u_1^2\}$ and $U_2 = \{u_2^1, u_2^2\}$ each have two possible instances, while object $U_3 = \{u_3^1, u_3^2, u_3^3\}$ has three possible instances. The probabilities of these instances is given as $P(u_1^1) = P(u_1^2) = 0.5$, $P(u_2^1) = 0.7$, $P(u_2^2) = 0.2$, $P(u_3^1) = 0.5$, $P(u_3^2) = 0.3$, $P(u_3^3) = 0.2$. Note that object U_2 is the only object having existential uncertainty: With a probability of $1 - 0.7 - 0.2 = 0.1$ object U_2 does not exist at all. Assuming independence between spatial objects, the probability for the possible world where $U_1 = u_1^1$, $U_2 = u_2^1$ and $U_3 = u_3^1$ is given by applying Equation 2.2 to obtain the product $0.5 \cdot 0.7 \cdot 0.5 = 0.175$. All possible worlds spanned by $\mathcal{DB}$ are depicted in Figure 2.3. The probability of each possible world is shown in Table 2.1, including possible worlds where U_2 does not exist.*

Recall that a predicate can evaluate to either true or false on a crisp (non-uncertain) database. An exemplary predicate is *There are at least five database objects in a 500meter range of the location "Theresienwiese, Munich".* To evaluate a predicate ϕ on an uncertain database using possible world semantics, the query predicate is evaluated on each possible world. The probability that the query predicate evaluates to true is defined as the sum of probabilities of all worlds where ϕ is satisfied, formally:

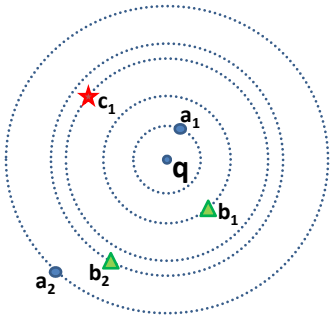
Definition 8. *Let $\mathcal{DB}$ be an uncertain spatial database inducing the set of possible worlds $\mathcal{W}$, let ϕ be some query predicate, and let*

$$\mathcal{I}(\phi, w \in \mathcal{W}) := P(\phi(\mathcal{DB}) | \mathcal{DB} = w) \in \{0, 1\}$$

be the indicator function that returns one if world w satisfies ϕ and zero otherwise. The marginal probability $P(\phi(\mathcal{DB}))$ of the event $\phi(\mathcal{DB})$ that predicate ϕ holds in $\mathcal{DB}$ is defined as using the theorem of total probability [220]:

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W}} \mathcal{I}(\phi, w) \cdot P(w) \tag{2.3}$$

The main challenge of analyzing uncertain data is to efficiently and effectively deal with the large number of possible worlds induced by an uncertain database $\mathcal{DB}$. In the case of continuous uncertain data, the number of possible worlds is uncountably infinite, even in the case where the database contains only one uncertain object. Thus, expensive integration operations or numerical approximation are required for most spatial database queries and spatial data mining tasks. Even in the case of discrete uncertainty, the number of possible worlds grows exponentially in the number of objects: in the worst case, any combination of alternatives of objects may have a non-zero probability, as shown exemplary in Figure 2.3. This large number of possible worlds makes efficient query processing and data mining an extremely challenging problem. In particular, any problem that requires an enumeration of all possible worlds is #P-hard. In particular, a number of probabilistic problems have been proven to be in #P [194]. Following this argumentation, general query processing in the case of discrete data using object independence has proven to be a #P-hard problem [57] in the context of relational data. The spatial case is a specialization of the relation case, but clearly, the spatial case is in #P as well, which becomes evident by construction of a query having an exponentially large result, such as the query that returns all possible worlds. Consequently, there can be no universal solution that allows to answer *any* query in polynomial time. This implies that querying processing on models that are generalizations of the discrete case with object independence, e.g., models using continuous distribution, or models that relax the object independence assumption, must also be a #P hard problem. The result of [57] implies that there exists query predicates, for which no polynomial time solution can be given. Yet, this result does not outrule the existence of query predicates that can be answered efficiently. For example the (trivial) query that always returns the empty set of objects can be efficiently answered on uncertain spatial databases. Now, the imminent question is whether the most important types of spatial queries can be answered efficiently. This thesis will positively answer this question, and propose exact and efficient solutions for the most prominent spatial query types. Furthermore, this thesis will present a paradigm for query processing and data mining on uncertain spatial data that can be adapted to new query types and data mining tasks.



P(a1)=0.1
P(a2)=0.9
P(b1)=0.6
P(b2)=0.4
P(c1)=1.0

World	Rank 1	Rank 2	Rank 3	P(w)
$w_1=a_1\,b_1\,c_1$	A	B	C	0.04
$w_2=a_1\,b_2\,c_1$	A	C	B	0.06
$w_3=a_2\,b_1\,c_1$	B	C	A	0.54
$w_3=a_2\,b_2\,c_1$	C	B	A	0.36

Figure 2.4: Example Database showing possible positions of uncertain objects and their corresponding probabilities.

2.4 Probabilistic Answer Semantics

Example 7. *Recall that a spatial similarity queries always requires a query object q and, informally speaking, returns objects to the user that are similar to q. In the case of uncertain data, there exists two fundamental semantics to describe the result of such a probabilistic spatial similarity query. These different answer semantics will be denoted as* object based answer semantics *and the* result based answer semantics*. Informally, the former semantics return possible* result objects *and their probability of being part of the result, while the later semantics return possible results, which consist of a single object, of a set of objects or of a sorted list of objects depending on the query predicate, and their probability of being the result as a whole.*

To the best of my knowledge, the only publication where this classification been presented is our VLDB tutorial [157] in the context of probabilistic kNN queries on uncertain data. There exists no publication explicitly explaining these semantics in the general case. All existing publications on uncertain data implicitly assume either semantics. I feel that the explicit identification and definition of these semantics is a necessary step to put existing work into context.

2.4.1 Object Based Probabilistic Answer Semantics

Using object based probabilistic answer semantics*, a probabilistic spatial query returns a set of objects, each associated with a probability describing the individual likelihood of this object to satisfy the spatial query predicate.*

Definition 9 (Object Based Answer Semantics)**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let q be a query object and let ϕ denote a spatial query predicate. Under object based (OB) probabilistic answer semantics, the result of a probabilistic spatial ϕ query is a set $\phi_{OB}(q,\mathcal{DB}) = \{(o \in \mathcal{DB}, P(o \in \phi_{OB}(q,\mathcal{DB})))\}$ of pairs. Each pair consists of a result object o and its probability $P(o \in \phi_{OB}(q,\mathcal{DB}))$ to satisfy ϕ. Applying possible world semantics*

(c.f. Definition 7) to compute the probability $P(o \in \phi_{OB}(q, \mathcal{DB}))$ *yields*

$$P(o \in \phi_{OB}(q, \mathcal{DB})) = \sum_{w \in \mathcal{W}, o \in \phi(q,w)} P(w), \tag{2.4}$$

where $\phi(q, w)$ *is the deterministic result of a spatial* ϕ *query having query object* q *applied to the deterministic database defined by world* w*.*

Formally, the result of a probabilistic spatial query under object based answer semantics is a function

$$\phi_{OB}(q, \mathcal{DB}) : \mathcal{DB} \rightarrow [0, 1]$$
$$o \mapsto P(o \in \phi_{OB}(q, \mathcal{DB})).$$

mapping each object o *in* $\mathcal{DB}$ *(the results) to a probability value.*

Figure 2.4 depicts a database containing objects $\mathcal{DB} = \{A, B, C\}$*. Objects* A *and* B *have two alternative locations each, while the position of* C *is known for certain. The locations and the probabilities of all alternatives are also depicted in Figure 2.4. This leads to a total number of four possible worlds. For example, in world* w_1 *where* $A = a_1$*,* $B = b_1$ *and* $C_1 = c_1$*, object* A *is closest to* q*, followed by objects* B *and* C*. Assuming inter-object independence, the probability of this world is given by the product of individual instance probabilities* $P(w_1) = P(a_1) \cdot P(b_1) \cdot P(c_1) = 0.04$*. The ranking of each possible world and the corresponding probability is also depicted in Figure 2.4. For a probabilistic* $2NN$ *query for the depicted query object* q*, the object based answer semantic computes the probability of each object to be in the two-nearest neighbor set of* q*. For object* A*, the probability* $P(A)$ *of this event equals* 0.1*, since there exists exactly two possible worlds* w_1 *and* w_2 *with a total probability of* $0.04 + 0.06 = 0.1$ *in which* A *is on rank one or on rank two, yielding a result tuple* $(A, 0.1)$*. The complete result of a* $P2NN$ *query under object based answer semantic is* $\{(A, 0.1), (B, 0.94), (C, 0.96)\}$*. Note that in general, objects having a zero probability are included in the result. For instance, assume an additional object* D *such that all instances of* D *have a distance to* q *greater than the distance between* q *and* b_2*. In this case, the pair* $(D, 0)$ *would be part of the result.*

The result of a query under object based probabilistic answer semantics contains one result tuple for every single database object, even if the probability of the corresponding object to be a result is very low or zero. In many applications, such results may be meaningless. Therefore, the size of the result set can be reduced by using a probabilistic query predicate as explained later in Section 2.5. A computational problem is the computation of the probability $P(o \in \mathcal{DB})$ of an object o to satisfy the spatial query predicate. In the example, this probability was derived by iterating over the set of all possible worlds $w_1, ..., w_4$. Since this set grows exponentially in the number of objects, such an approach is not viable in practice. Therefore, efficient techniques to compute the probability values $P(o)$ are required. Such techniques must avoid an explicit enumeration of all possible worlds and will be presented for various query predicates in Part III of this thesis.

This thesis will apply object based answer semantics, following the general trend of the a majority of related research. Nevertheless, to understand some of the remaining related

work, result based answer semantics will be explained in the following. To understand the different of both result semantics is essential: in some related publication the problem of answering some probabilistic query may be proven to be in $\#P$, while another related publication gives a solution problem that lies in P-TIME, even though both the spatial query predicate and the probabilistic query predicate are identical. In such cases, different answer semantics may explain these results without assuming $P = NP$.

2.4.2 Result Based Probabilistic Answer Semantics

In the case of result based answer semantics, possible result sets of a probabilistic spatial query are returned, each associated with the probability of this result.

Definition 10 (Result Based Answer Semantics)**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let q be a query object and let ϕ denote a spatial query predicate. Under result based (RB) answer semantics, the result of a probabilistic spatial ϕ query is a set*

$$\phi_{RB}(q, \mathcal{DB}) = \{(r, P(r)) | r \subseteq \mathcal{DB}, P(r) = \sum_{w \in \mathcal{W}, \phi(q,w)=r} P(w)\}$$

of pairs. This set contains one pair for each result $r \subseteq \mathcal{DB}$ associated with the probability $P(r)$ of r to be the result. Following possible world semantics, the probability $P(r)$ is defined as the sum of probabilities of all worlds $w \in \mathcal{W}$ such that a spatial ϕ query returns r.

Formally, the result of a probabilistic spatial query under result based answer semantics is a function

$$\phi_{RB}(q, \mathcal{DB}) : \overline{\mathcal{P}}(\mathcal{DB}) \to [0, 1]$$
$$r \mapsto P(r).$$

mapping a elements of the power set $\overline{\mathcal{P}}(\mathcal{DB})$ (the results) to probability values.

Example 8. *For a probabilistic $2NN$ query for the depicted query object q, result based answer semantics require to compute the probability of each subset of $\{A, B, C\}$ to be in the two-nearest neighbor set of q. For the set $\{B, C\}$, the probability of this event is 0.90, since there is two possible worlds w_3 and w_4 with a total probability of $0.54 + 0.36 = 0.9$ in which B and C are both contained in the $2NN$ set of q. Note that in worlds w_3 and w_4 objects B and C appear in different ranking positions. This fact is ignored by a kNN query, as the results are returned unsorted. In this example, the complete result of a $P2NN$ query under object based answer semantic is $\{(\{A, B, C\}, 0), (\{A, B\}, 0.04), (\{A, C\}, 0.06), (\{B, C\}, 0.90), (\{A\}, 0), (\{B\}, 0), (\{C\}, 0), (\{\emptyset\}, 0)\}$.*

Clearly, the result of a query using result based answer semantics can be used to derive the result of an identical query using object based answer semantics. For instance, the result of Example 8 implies that the probability of object A to be a $2NN$ of q is 0.10, since there exists two possible results using result based answer semantics, namely $(\{A, B\}, 0.04) and (\{A, C\}, 0.06)$ having a total probability of $0.04 + 0.06 = 0.1$, which matches the result of Example 7.

Lemma 1. *Let q be the query point of a probabilistic spatial τ query. It holds that the result of this query using object based answer semantics $\phi_{OB}(q, \mathcal{DB})$ is functionally dependent of the result of this query using result based answer semantics. The set $PS\phi Q_{OB}(q, \mathcal{DB})$ can be computed given only the set $PS\phi Q_{RB}(q, \mathcal{DB})$ as follows:*

$$PS\phi Q_{OB}(q, \mathcal{DB}) = \{(o, P(o)) | o \in \mathcal{DB} \wedge P(o) = \sum_{(r,P(r)) \in PS\phi Q_{RB}(q,\mathcal{DB}), o \in r} P(r)\}$$

Proof. Let $\mathcal{W}$ denote the set of possible worlds of $\mathcal{DB}$, and let $p(w \in \mathcal{W})$ denote the probability of a possible world. Furthermore, let

$$w_{S \subseteq \mathcal{DB}} := \{w \in \mathcal{W} | \phi(q, w) = S\}$$

denote the set of possible worlds such that $\phi(q, w) = S$, i.e., such that the predicate that a ϕ query using query object q returns set S holds. In each world w, query q returns exactly one deterministic result $PS\phi Q_{RB}(q, w)$. Thus, the sets $w_{S \subseteq \mathcal{DB}}$ represent a complete and disjunctive partition of $\mathcal{W}$, i.e., it holds that

$$\mathcal{W} = \bigcup_{S \subseteq \mathcal{DB}} w_S \tag{2.5}$$

and

$$\forall R, S \in \overline{\mathcal{P}}(\mathcal{DB}) : R \neq S \Rightarrow w_R \bigcap w_S = \emptyset. \tag{2.6}$$

Using Equations 2.5 and 2.6, we can rewrite Equation 2.4

$$P(o \in \phi_{OB}(q, \mathcal{DB})) = \sum_{w \in \mathcal{W}, o \in \phi(q,w)} P(w)$$

as

$$P(o \in \phi_{OB}(q, \mathcal{DB})) = \sum_{S \in \overline{\mathcal{P}}(\mathcal{DB})} \sum_{w \in w_S, o \in \phi(q,w)} P(w).$$

By definition, query q returns the same result for each world in $w \in w_S$. This result contains object o if $o \in S$. Thus we can rewrite the above equation as

$$P(o) = \sum_{S \in \overline{\mathcal{P}}(\mathcal{DB}), o \in S} P(S).$$

The probabilities $P(S)$ are given by function $PS\phi Q_{RB}(q, \mathcal{DB})$. □

In the above proof, we have performed a linear-time reduction of the problem of answering probabilistic spatial queries using object based answer semantics to the problem of answering probabilistic spatial queries using result based answer semantics. Thus, we have shown that, except for a linear factor (which can be neglected for most probabilistic spatial query types, since most algorithm run in no better than log-linear time), the problem of answering a probabilistic spatial query using result based answer semantics is at least as hard as answering a probabilistic spatial query using object based semantics.

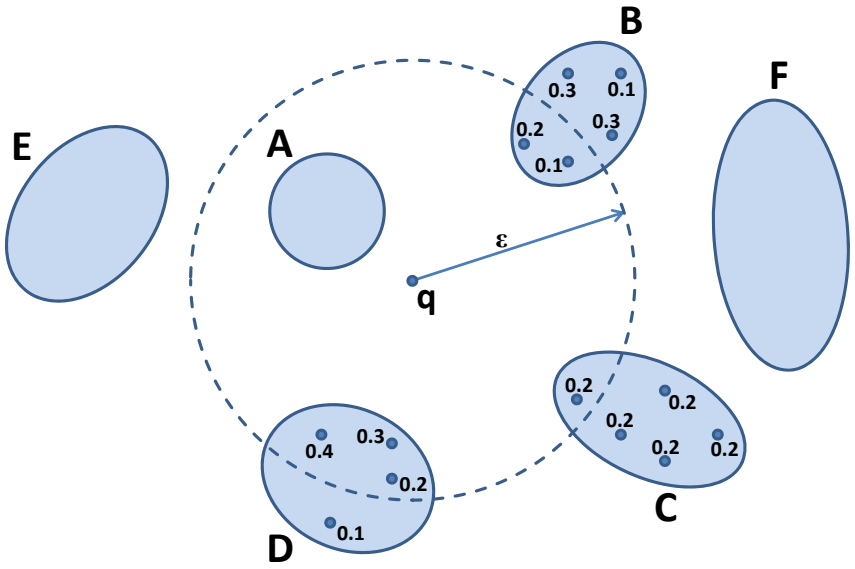


Figure 2.5: Example of an uncertain ϵ-range query. Object A is a true hit, objects B, C and D are possible hits.

2.5 Probabilistic Query Predicates

As explained in Section 1.1, a spatial similarity query requires to evaluate spatial query predicates such as *the distance between two objects is less than* ϵ and *one object is the nearest neighbor of another object*. Generally, in an uncertain database, the question whether an object satisfies a given query predicate ϕ cannot be answered deterministically, due to possible uncertainty of object attributes. Due to this uncertainty, the predicate that an object satisfies ϕ is a random variable, having some (possibly zero, possibly one) probability. A probabilistic query predicate quantifies the minimal probability required for a result to qualify as a result that is sufficiently significant to be returned to the user. This section formally define probabilistic query predicate for general query predicates. The following definition are made for uncertain data in general, but can be applied analogously for uncertain spatial data.

A *probabilistic query* can be defined without any probabilistic query predicate. In this case, all objects, and their respective probabilities are returned.

Definition 11 (Probabilistic Query)**.** *Let $\mathcal{DB}$ be an uncertain database, let q be a query point and let ϕ be a query predicate. A* probabilistic query $\phi(q, \mathcal{DB})$ *returns all database objects $o \in \mathcal{DB}$ together with their respective probability $P(o \in \phi(q, \mathcal{DB}))$ that o satisfies ϕ.*

$$\phi(q, \mathcal{DB}) = \{(o \in \mathcal{DB}, P(o \in \phi(q, \mathcal{DB})))\} \tag{2.7}$$

The term *probabilistic* query is simply derived from the fact that unlike a traditional query, a probabilistic query result has probability values associated with each result. The main challenge of answering a probabilistic query, is to compute the probability $P(o \in$

$\phi(q, \mathcal{DB})$) for each object. Using possible world semantics, a probabilistic query can be answered by evaluating the query predicate for each object and each possible world, i.e.,

$$P(o \in \phi(q, \mathcal{DB})) := \sum_{w \in \mathcal{W} f_{ind}(\phi, w) \cdot P(w)} .$$

But clearly, it is necessary to avoid the combinatorial growth that would be induced by this "naive" evaluation method.

Example 9. *For example, consider the query "Return all friends of user q having a spatial distance of less than* $100m$ *to q" depicted in Figure 2.5. Thus, the predicate* ϕ *is a* $100m$*-range predicate using query point q. We can deterministically tell that friend A must be within* $\epsilon = 100m$ *Euclidean distance of q, while friends E and F cannot possibly be in range. The pairs* $(A, 1)$*,* $(E, 0)$ *and* $(F, 0)$ *are added to the result. For friends B, C and D, this predicate cannot be answered deterministically. Here, friend B has some possible positions located inside the* $100m$ *range of q, while other possible positions are outside this range. The two locations inside q's range have a probability of* 0.1 *and* 0.2*, respectively, thus the total probability of object B to satisfy the query predicate is* $0.1 + 0.2 = 0.3$*. The pair* $(B, 0.3)$ *is thus added to the result. The pairs* $(C, 0.2)$ *and* $(D, 0.9)$ *complete the result* $100m\text{-}range(q, \mathcal{DB}) = \{(A, 1), (B, 0.3), (C, 0.2), (D, 0.9), (E, 0), (F, 0)\}$.

The immediate question in the above example is:"Is a probability of 0.3 sufficient to warrant returning B as a result?". To answer this question, a probabilistic query can explicitly specify a probabilistic query predicate, in order to return only significant results having a sufficiently high probability.

2.5.1 Probabilistic Threshold Queries

This paragraph defines a probabilistic query predicate that allows to return only results that are statistically significant.

Definition 12 (Probabilistic Threshold Query (PτQ))**.** *Let* $\mathcal{DB}$ *be an uncertain (spatial) database, let q be a spatial query object, let* $0 \leq \tau \leq 1$ *be a real value and let* ϕ *be a spatial query predicate. A* probabilistic τ query *(PτQ) returns all objects* $o \in \mathcal{DB}$ *such that o has a probability of at least* τ *to satisfy* $\phi(q, \mathcal{DB})$*:*

$$P\tau\phi(q, \mathcal{DB}) := \{o \in \mathcal{DB} | P(o \in \phi(q, \mathcal{DB})) \geq \tau\}.$$

Example 10. *In Figure 2.5, a probabilistic threshold* $100m\text{-}range(q,\mathcal{DB})$ *query with* $\tau = 0.5$ *query returns the set of objects* $P0.5\,100m\text{-}range(q, \mathcal{DB}) = \{A, D\}$*, since objects A and D are the only objects such that their total probability of alternatives inside the query region is equal or greater to* $\tau = 0.5$.

Semantically, a probabilistic threshold spatial query returns all results having a statistically significant probability to satisfy the query predicate. Therefore, the probabilistic

threshold query serves as a statistical test of the hypothesis "o is a result" at a significance level of τ. This test uses the probability $P(o \in \phi(q, \mathcal{DB}))$ as a test statistic. Efficient algorithms to compute this probability $P(o \in \phi(q, \mathcal{DB}))$, will be presented in Part III for the cases of ϵ-range queries, kNN queries, similarity ranking queries and RkNN queries.

A probabilistic threshold query on uncertain spatial data is useful in applications, where the parameters of the spatial predicate τ (e.g. the range of an ϵ-range query, or the parameter k of a kNN query), as well as the probabilistic threshold τ are chosen wisely, requiring expert knowledge about the database $\mathcal{DB}$. If these parameters are chosen inappropriately, no results may be returned, or the set of returned result may grow too large. For example, if τ is chosen very large, and if the database has a high grade of uncertainty, then no result may be returned at all. Analogously, if the parameter ϵ is chosen too small then no result will be returned, while a too large value of ϵ may return all objects.

2.5.2 Probabilistic Top$k$ Queries

In cases where insufficient information is given to select appropriate parameter values, the following probabilistic query predicate is defined to guarantee that only the k most significant results are returned.

Definition 13 (Probabilistic Top$k$ Query (PTop$k$Q))**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let q be a spatial query object, let $1 \leq k \leq |\mathcal{DB}$ be a positive integer, and let ϕ be a spatial query predicate. A* probabilistic spatial Top$k$ query *(PTop$k$Q) returns the smallest set PTop$k\phi$(q,$\mathcal{DB}$) of at least k objects such that*

$$\forall U_i \in PTopk\phi(q, \mathcal{DB}), U_j \in \mathcal{DB} \setminus PTopk\phi(q, \mathcal{DB}) : P(U_i \in \phi(q, \mathcal{DB})) \geq P(U_j \in \phi(q, \mathcal{DB}))$$

Thus, a probabilistic spatial Top$k$ query returns the k objects having the highest probability to satisfy the query predicate. Again, in case of ties, the resulting set may be greater than k.

Example 11. *In Figure 2.5, a PTop3 ϕ query using a $\phi = 100m$-range spatial predicate returns objects PTop3 $100m$-range$(q, \mathcal{DB}) = \{A, B, D\}$, since these objects have the highest probability to satisfy the spatial predicate, i.e., have the highest probability to be located in the spatial $100m$-range.*

Note, that the probabilistic Top$k$ query predicate can be combined with a kNN spatial query, i.e., with the case where $\phi = kNN$. Such a probabilistic Top$k$ jNN query returns the set of k objects having the highest probability, to be j-nearest neighbor of the query object. Clearly, k and j may have different integer values, such that differentiation is needed.

2.5.3 Discussion

In summary, a probabilistic spatial query is defined by two query predicates:

- A spatial predicate ϕ to select uncertain objects having sufficiently *high proximity* to the query object, and
- a probabilistic predicate ψ, to select uncertain objects having sufficiently *high probability* to satisfy ϕ.

It has to be mentioned, that alternatively to this definition, a single predicate can be used, that combines both spatial and probabilistic features. For example, a monotonic score function can be utilized, which combines spatial proximity and probability to return a single scalar score. An example of such a monotone score function is the expected distance function

$$E(dist(q, U \in \mathcal{DB})) = \sum_{u \in U} P(u) \cdot dist(q, u),$$

where q is the query object, and $\mathcal{DB}$ is an uncertain database. The expected support function is utilized by a number of related publications, such as [133, 55]. Using such a monotone score function, objects with a sufficiently high score can be returned. The advantage of using such an approach, is that objects that are located very close to the query require a lower probability to be returned as a result, while objects that are located further away from the query object require a higher probability. Yet, the main problem of such a combined predicate, is that the probability of an object is treated as a simple attribute, thus losing its probabilistic semantic. Thus, the resulting score is very hard to interpret. An object that has a high score, may indeed have a very low probability to exist at all, because it is located (if it exists) very close to the query object. Consequently, the score itself no longer contains any confidence information, and thus, it is not possible to answer queries according to possible world semantics using a single aggregate, such as expected distance, only.

2.6 Approximate Queries

In general, the problem of probabilistic query processing on uncertain data has been proven to be in $\#P$ [57]. Thus, there exists query predicates that require to consider an exponential large set of cases. For such queries, an analytical solution is infeasible. Furthermore, even queries that can be answered in $PTIME$ may be too slow in practice. Yet, exact result probabilities are not required in many applications. A good approximation of result probabilities with probabilistic guarantees may suffice in such applications.

2.6.1 Monte Carlo Algorithms

Using object based query semantics, each object $o \in \mathcal{DB}$ has a probability $P(o \in \phi(\mathcal{DB})) \in [0, 1]$ to satisfy the spatial query predicate. Thus, the event that o satisfies ϕ is a binomial

random variable, having a probability of $P(o \in \phi(\mathcal{DB}))$ to be true, and a probability of $1-P(o \in \phi(\mathcal{DB}))$ to be false. To estimate the a-priori unknown probability $P(o \in \phi(\mathcal{DB}))$, random worlds can be instantiated. For a given possible world $w \in \mathcal{W}$, a traditional (non-uncertain) ϕ query can be executed to decide whether $o \in \phi(q, w)$.

Definition 14 (Monte-Carlo Approach)**.** *Let $P\psi\phi(q,\mathcal{DB})$ be a spatial query, where $\mathcal{DB}$ is an uncertain spatial database, q is a query point, ϕ is a spatial query predicate and ψ is a probabilistic query predicate. A* Monte-Carlo *approach creates a multi-set $\mathcal{S} = \{w_1, ..., w_{|\mathcal{S}|}\}, w_i \in \mathcal{W}$ of randomly and independently sampled possible worlds.*[1] *The spatial query predicate $\phi(q, w)$ is evaluated on each possible world. The estimator*

$$\hat{\mu} := \frac{\sum_{w \in \mathcal{S}} f_{ind}(o \in \phi(q, w))}{|\mathcal{S}|}, \tag{2.8}$$

yields an unbiased estimator of the probability $P(o \in \phi(q, \mathcal{DB}))$, where $f_{ind}(o \in \phi(q, w))$ is an indicator function that returns one if $o \in \phi(q, w)$ and zero otherwise.

Note that $\hat{\mu}$ is a random variable, since each world $w \in S$ is chosen at random, each having a probability of $P(o \in \phi(q, \mathcal{DB}))$ to satisfy ϕ. In particular $\hat{\mu}$ is the relative fraction of successful Bernoulli trials out of a total of $|\mathcal{S}|$ Bernoulli trials, each having a success probability of $P(o \in \phi(q, \mathcal{DB}))$. Consequently, $\hat{\mu}$ follows a Binomial $B(|\mathcal{S}|, P(o \in \phi(q, \mathcal{DB})))$ distribution.

Since the number of possible worlds increases exponential in the number of uncertain objects in $\mathcal{DB}$, any probabilistic ϕ query algorithm that explicitly considers each possible world individually will be inapplicable to handle databases of non-trivial size. A Monte-Carlo algorithm entirely avoids to use any expensive probabilistic query processing algorithm. Instead, any non-uncertain ϕ query processing algorithm can be utilized to compute the indicator function $f_{ind}(o \in \phi(q, w))$. This has to be repeated a total of $|\mathcal{S}|$ times leading to a total time complexity of $O(|S| \cdot \rho)$, where ρ is the time complexity of the problem of answering a spatial ϕ query on (certain) point data.

Lemma 2. *If the sample of possible worlds S is drawn in an unbiased way, then the Monte-Carlo estimator $\hat{\mu}$ is unbiased, i.e.,*

$$E(\hat{\mu}) = P(o \in \phi(q, \mathcal{DB})),$$

where $P(o \in \phi(q, \mathcal{DB}))$ is the probability that an object o is in the result of a spatial query predicate ϕ having query object q on database $\mathcal{DB}$.

Proof. Using Definition 14 we get

$$E(\hat{\mu}) = E(\frac{\sum_{w \in \mathcal{S}} f_{ind}(o \in \phi(q, w))}{|\mathcal{S}|}).$$

[1] This implies that the same world may be sampled in $\mathcal{S}$ more than once. Thus $\mathcal{S}$ is defined as a multi-set rather than a set, to allow duplicate elements.

Exploiting linearity of expectation we get

$$E(\hat{\mu}) = \sum_{w \in \mathcal{S}} \frac{E(f_{ind}(o \in \phi(q, w)))}{|\mathcal{S}|} \tag{2.9}$$

By definition of the expected value as $E(X) = \sum_x x \cdot P(x)$, and by assuming that any world w is sampled randomly without any bias we get

$$E(f_{ind}(o \in \phi(q, w)) = \sum_{w \in \mathcal{W}} f_{ind}(o \in \phi(q, w)) \cdot P(w)$$

By definition of possible world semantics (Definition 8), it holds that the sum on the right hand side of the above equation is equal to $P(o \in \phi(q, \mathcal{DB}))$. This yields

$$E(f_{ind}(o \in \phi(q, w)) = P(o \in \phi(q, \mathcal{DB})) \tag{2.10}$$

Substitution of Equation 2.10 in Equation 2.9 yields

$$E(\hat{\mu}) = \sum_{w \in \mathcal{S}} \frac{P(o \in \phi(q, \mathcal{DB}))}{E(|\mathcal{S}|)}$$

Since the expected value $E(|\mathcal{S}|)$ of a constant $|\mathcal{S}|$ equals $|\mathcal{S}|$ we get

$$E(\hat{\mu}) = \sum_{w \in \mathcal{S}} \frac{P(o \in \phi(q, \mathcal{DB}))}{|\mathcal{S}|}$$

Since the body $\frac{P(o \in \phi(q, \mathcal{DB}))}{|\mathcal{S}|}$ of the above sum no longer contains the parameter w, this body is added once for each iteration of the sum. As the sum is iterated exactly $|\mathcal{S}|$ times (once for each element of $\mathcal{S}$) we obtain

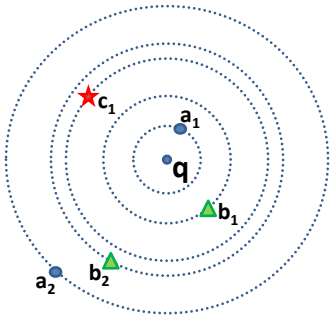
$$E(\hat{\mu}) = |\mathcal{S}| \frac{P(o \in \phi(q, \mathcal{DB}))}{|\mathcal{S}|}$$

which simplifies to

$$E(\hat{\mu}) = P(o \in \phi(q, \mathcal{DB})).$$

□

Example 12. *Once again, consider the example in Figure 2.6. As we have seen in Example 7, the probability of object A to be the result of a 2NN spatial query, is* 0.1*. This result was derived by explicitly considering all possible worlds, and thus, is not applicable for large databases. Using a Monte-Carlo approach, we draw a number of* $|\mathcal{S}| = 1000$ *sample worlds, getting a total of* 94 *worlds where A is in the 2NN result. This yields an estimator of* $\hat{\mu} = \frac{92}{1000} = 0.092$*, having an approximation error of only* 0.008*, which may be "good enough" in practice.*



P(a_1)=0.1
P(a_2)=0.9
P(b_1)=0.6
P(b_2)=0.4
P(c_1)=1.0

World	Rank 1	Rank 2	Rank 3	P(w)
w_1=a_1 b_1 c_1	A	B	C	0.04
w_2=a_1 b_2 c_1	A	C	B	0.06
w_3=a_2 b_1 c_1	B	C	A	0.54
w_3=a_2 b_2 c_1	C	B	A	0.36

Figure 2.6: Example Database showing possible positions of uncertain objects and their corresponding probabilities.

2.6.2 Probabilistic Guarantees

Whether a sample size is "good enough" is hard to assess in practice where the true value of $P(o \in \phi(q, \mathcal{DB})$ is not known. In the above example, there is a non-zero (albeit small) probability that out of 1000 worlds, there exists no world where A is a 2NN result. This would yield an estimator of $\hat{\mu} = 0$ which is quite far from the true value of $P(P(o \in \phi(q, \mathcal{DB})) = 0.1$. To ensure that the error is small, probabilistic guarantees are required.

Definition 15. *A probabilistic guarantee ensures that the error of an approximate technique does not exceed some error threshold ϵ with a probability of at least τ, i.e.,*

$$P(|\hat{\mu} - \mu| \leq \epsilon) \geq \tau,$$

where $\hat{\mu}$ is an estimator of a parameter μ.

In the case of Monte-Carlo estimation of the probability that an object $o \in \mathcal{DB}$ satisfies some spatial query predicate ϕ, we can exploit that by Definition 14 the estimator $\hat{\mu}$ follows a Binomial $B(|\mathcal{S}|, P(o \in \phi(q, \mathcal{DB})))$ distribution. This observation allows to apply Hoeffding's inequality [90] to obtain the following probabilistic guarantee:

$$P(|\hat{\mu} - P(o \in \phi(q, \mathcal{DB}))| < \epsilon) \geq 1 - 2e^{-2\epsilon^2|\mathcal{S}|}, \tag{2.11}$$

where $\hat{\mu}$ is is defined by Equation 2.8.

Example 13. *Reconsider the example in Figure 2.6 where the true probability $P(A)$ of object A to be the result of a 2NN spatial query, is* 0.1*. Assume that the number of Monte Carlo samples $|\mathcal{S}|$) is* 10,000*. According to Equation 2.11, the probability that the estimation error of $\hat{\mu}$ is less that* 0.01 *is at least*

$$P(|\hat{\mu} - P(P(o \in \phi(q, \mathcal{DB}))| < 0.01) \geq 1 - 2e^{-2 \cdot 0.01^2 \cdot 10000}$$

$$= 1 - 2 \cdot e^{-2} = 1 - 2 \cdot 0.135 = 0.73.$$

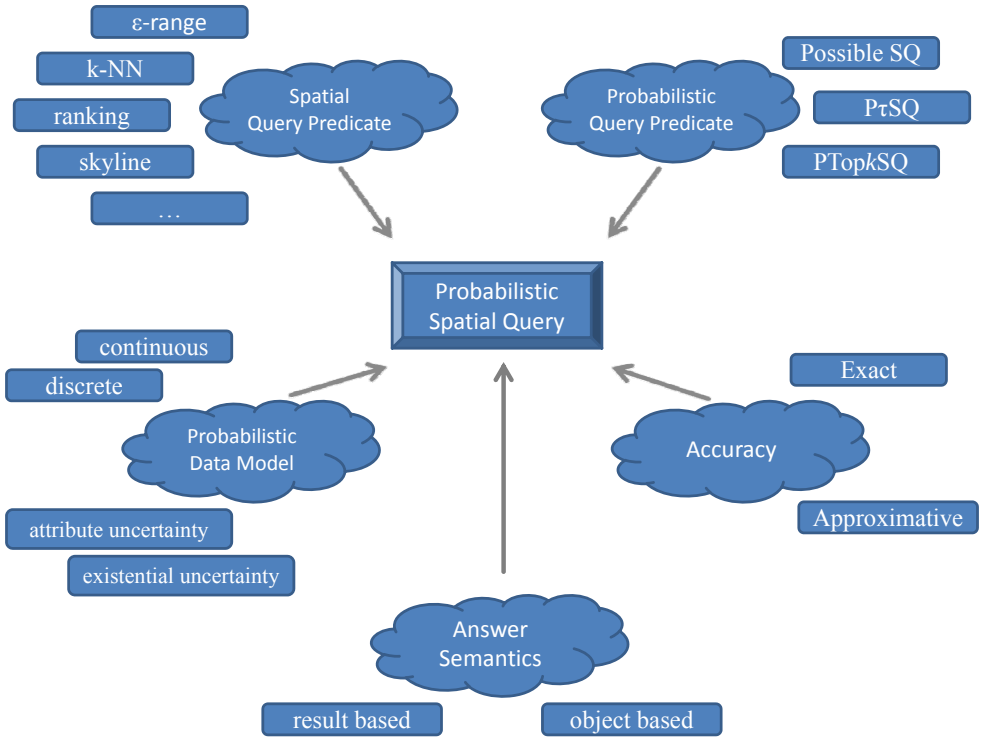


Figure 2.7: Components of a probabilistic spatial query.

Note that this approximation is not very tight. For example, if $|\mathcal{S}| = 1000$*, then Equation 2.11 yields*

$$P(|\hat{\mu} - P(o \in \phi(q, \mathcal{DB}))| < 0.01) \geq 1 - 2e^{-2 \cdot 0.01^2 \cdot 1000}$$
$$= 1 - 2 \cdot e^{-0.2} = 1 - 2 \cdot 0.819 = -0.637$$

which is a trivial statement.

While the bounds acquired by Equation 2.11 are rather loose, they can be evaluated very efficiently. Most importantly, these bounds allow to avoid evaluating the exact pdf of the binomial distribution $B(|\mathcal{S}|, P(P(o \in \phi(q, \mathcal{DB}))))$, which requires to evaluate large binomial coefficients.

2.7 Summary

A probabilistic spatial query is defined by a series of components which have been described in this section and which are summarized in Figure 2.7. These components include the choice of a spatial query predicate as well as a probabilistic query predicate. Both query predicates specify the requirements that objects must meet in order to be returned as a

result. Further, probabilistic answer semantics must be specified, to define whether result probabilities correspond to result sets, or to individual objects. Next, a probabilistic data representation must be chosen to model the data. Uncertainty models impose assumptions on the physical world in order to reduce computation complexity. The model of choice depends on the application and the assumption that can be made without vital loss of information. Based on the chosen model, a choice has to be made between analytical solutions to compute exact results, and numerical solutions that return approximate results. The choice of each individual component has a strong impact on the semantic and the complexity of a probabilistic spatial query.

In the subsequent parts of this work, the main contributions of this thesis on the field of uncertain spatial database research are presented:

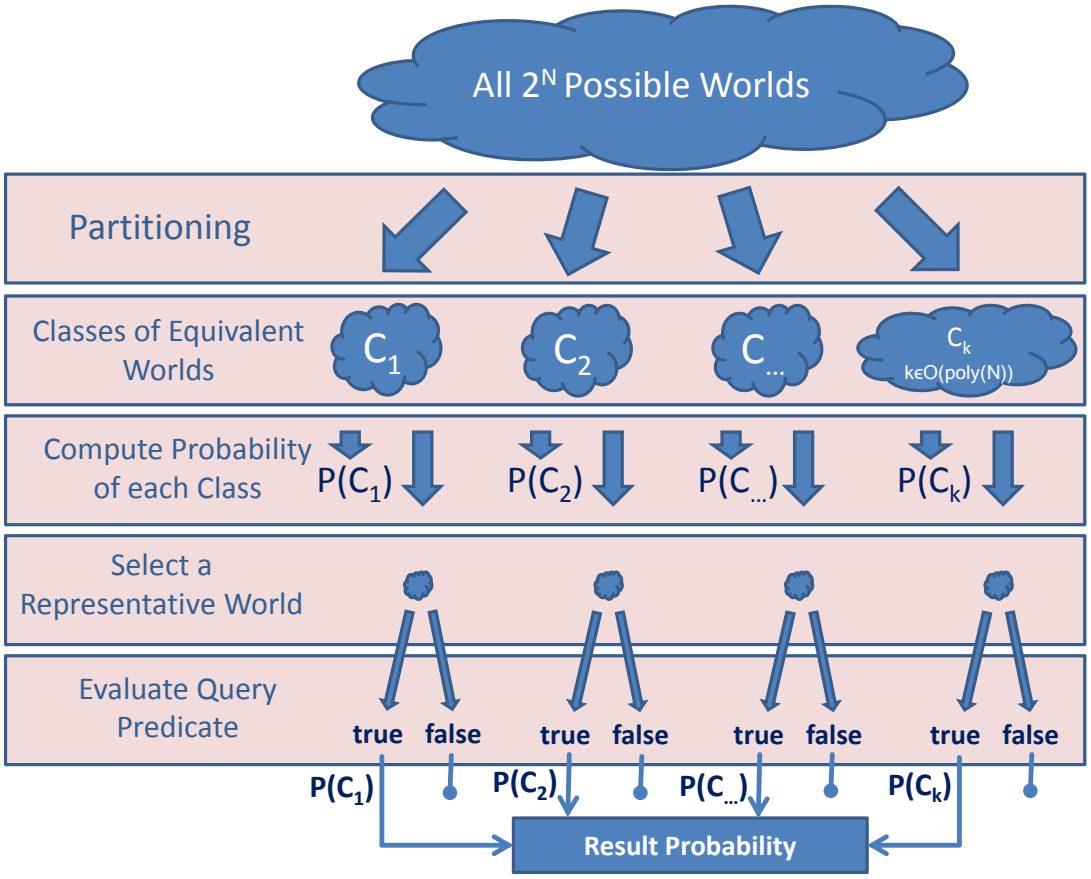
- First, a general paradigm for query processing on uncertain data is presented in Chapter 3. This paradigm will be applied throughout this thesis to find efficient solutions for the most important types of spatial queries and data mining tasks on uncertain spatial and spatio-temporal data.

- Based on this paradigm, exact and efficient solutions for answering probabilistic spatial queries on uncertain spatial databases are presented in Part III, including
  - probabilistic spatial range and window queries,
  - probabilistic k-nearest neighbor (kNN) queries,
  - probabilistic ranking queries and
  - probabilistic reverse k-nearest neighbor queries.

- Following the same paradigm, a solution to the spatial data mining problem of probabilistic spatial collocation mining on uncertain spatial databases is presented in Part IV.

Chapter 3

The Paradigm of Equivalent Worlds



In Section 2.3 the concept of possible world semantics has been introduced. Possible world semantics give an intuitive and mathematically sound interpretation of an uncertain spatial database. Furthermore, queries that adhere to possible world semantics return unbiased results, by evaluating the query on each possible world. Since any such approach requires to run queries on an exponential number of worlds, any naive approach is infeasible. Alternatively, we have seen that a Monte-Carlo approach (c.f. Section 2.6.1), which randomly samples possible worlds, will converge to the result of a query using possible worlds semantics, if the number of Monte-Carlo samples approaches infinity. Depending on the precision required by an application, a large number, ranging from thousands to millions of sample worlds have to be evaluated. This incurs significant computational effort and, at the same time, yields results having a possibly significant error.

3.1 Equivalent Worlds

The drawbacks of both naive and Monte-Carlo approaches can be completely avoided, if we can find a way to efficiently compute exact probabilities, while still adhering to possible world semantics. Introducing a general paradigm to achieve is goal, is the purpose of this part. Therefore, reconsider Definition 8, defining the probability that some predicate ϕ is satisfied in an uncertain database $\mathcal{DB}$ as the total probability of all possible worlds satisfying ϕ. Recall Equation 2.3

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W}} \mathcal{I}(\phi, w) \cdot P(w),$$

where $\mathcal{W}$ is the set of all possible worlds; $\mathcal{I}(\phi, w)$ is an indicator function that returns one if predicate ϕ holds (i.e., resolves to true) in the crisp database defined by world w and zero otherwise, and $P(w)$ is the probability of world w. To reduce the number of possible worlds that need to be considered to compute $P(\phi(\mathcal{DB}))$, we first need the following definition.

Definition 16 (Class of Equivalent Worlds)**.** *Let ϕ be a query predicate and let $S \subseteq \mathcal{W}$ be a set of possible worlds such that for any two worlds $w_1, w_2 \in S$ we can guarantee that ϕ holds in world w_1 if an only if ϕ holds in world w_2, i.e.,*

$$\forall w_1, w_2 \in S : \mathcal{I}(\phi, w_1) \Leftrightarrow \mathcal{I}(\phi, w_2)$$

Then set S is called a class of worlds equivalent with respect to ϕ. *In the remainder of this thesis, if the spatial query predicate ϕ is clearly given by the context, then S will simply be denoted as a* class of equivalent worlds. *Any worlds $w_i, w_j \in S$ are denoted as* equivalent worlds.

We now make the following observation:

Corollary 1. *Let $S \subseteq \mathcal{W}$ be a class of worlds equivalent with respect to ϕ (c.f. Definition 16, we can rewrite Equation 2.3 as follows:*

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W}} \mathcal{I}(\phi, w) \cdot P(w) \Leftrightarrow$$

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W} \setminus S} \mathcal{I}(\phi, w) \cdot P(w) + \mathcal{I}(\phi, w \in S) \cdot \sum_{w \in S} P(w). \tag{3.1}$$

Proof. Due to the assumption that for any two worlds $w_1, w_2 \in S$ it holds that ϕ holds in world w_1 if an only if ϕ holds in world w_2, we get $\mathcal{I}(\phi, w_1) = 1 \Leftrightarrow \mathcal{I}(\phi, w_2) = 1$ and $\mathcal{I}(\phi, w_1) = 0 \Leftrightarrow \mathcal{I}(\phi, w_2) = 0$ by definition of function $\mathcal{I}$. Due to this assumption, we have to consider two cases.

Case 1: $\forall w \in S : \mathcal{I}(\phi, w) = 0$

In this case, both Equation 2.3 and Equation 3.1 can be rewritten as

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W} \setminus S} \mathcal{I}(\phi, w) \cdot P(w).$$

Case 2: $\forall w \in S : \mathcal{I}(\phi, w) = 1$

In this case, both Equation 2.3 and Equation 3.1 can be rewritten as

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W} \setminus S} \mathcal{I}(\phi, w) \cdot P(w) + \sum_{w \in S} P(w)$$

□

The only different between both cases if the additive term $\sum_{w \in S} P(w)$, which exists only in Case 2. The indicator function $\mathcal{I}(\phi, w \in S)$ ensure that this term is only added in the second case. As main purpose, Corollary 1 states that, given a set of equivalent worlds S, we only have to evaluate the indictor function $\mathcal{I}(\phi, w)$ on a single representative world $w \in S$, rather than on each world in S. This allows to reduce the number of (crisp) ϕ queries required to compute Equation 2.3 by $|S| - 1$.

Corollary 1 leads to the following Lemma.

Lemma 3. *Let $\mathcal{S}$ be a partitioning of $\mathcal{W}$ into disjunctive sets such that $\bigcup_{S \in \mathcal{S}} S = \mathcal{W}$ and for all $S_1, S_2 \in \mathcal{S} : S_1 \cap S_2 = \emptyset$. Equation 2.3 can be rewritten as*

$$P(\phi(\mathcal{DB})) = \sum_{w \in \mathcal{W}} \mathcal{I}(\phi, w) \cdot P(w) \Leftrightarrow$$

$$P(\phi(\mathcal{DB})) = \sum_{S \in \mathcal{S}} \mathcal{I}(\phi, w \in S) \cdot \sum_{w \in S} P(w). \tag{3.2}$$

Proof. Lemma 3 is derived by applying Corollary 1 once for each $S \in \mathcal{S}$. □

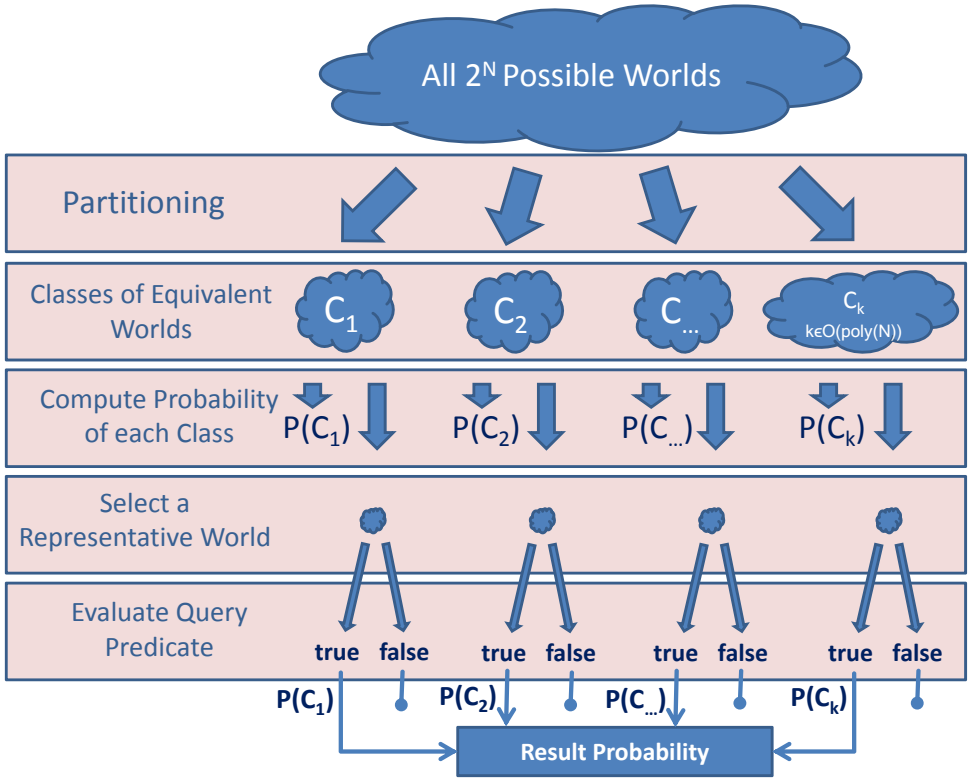


Figure 3.1: Summary of the Paradigm of Equivalent Worlds.

3.2 Exploiting Equivalent Worlds for Efficient Algorithms

Given a partitioning $\mathcal{S}$ of all possible worlds, Equation 3.2 requires to perform the following two tasks. The first task requires to evaluate the indicator function $\mathcal{I}(\phi, w \in S)$ for one representative world of each partition. This can be achieved by performing a traditional (non-uncertain) ϕ query on these representatives. The final challenge is to efficiently compute the total probability $P(S) := \sum_{w \in S} P(w)$ for each equivalent class $S \in \mathcal{S}$. This computation must avoid an enumeration of all possible worlds, i.e., must be in $o(|S|)$.[1] Achieving an efficient computation is a creative task, and usually requires to exploit properties of the model (such as object independence) and properties of the spatial query predicate. The paradigm of equivalent worlds is illustrated and summarized in Figure 3.1. In the first step, set of all possible worlds $\mathcal{W}$, which is exponential in the number N of uncertain objects, has to be partitioned into a polynomial large set of classes of equivalent worlds, such that all worlds in the same class are guaranteed to be equivalent given the query predicate ϕ. This yields a the set $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ of classes of equivalent worlds.

[1] Note that if an exponential large set is partitioned into a polynomial number of subsets, then at least on such subset must have exponential size. This is evident considering that $O(\frac{2^n}{poly(n)}) = O(2^n)$.

To allow efficient processing, this set must be polynomial in size, since each class has to be considered individually in the following. Next, we require to compute the probability of each class C_i, without enumeration of all possible worlds contained in C_i, the number of which may still be exponential. In fact, at least one class C_i must contain $O(2^N)$ possible worlds. Next, we need to decide, for each class C_i, whether all worlds $w \in C_i$ satisfy the query predicate ϕ, or whether no world $w \in C_i$ satisfies ϕ. Due to equivalence of all possible worlds in C_i, these are the only possible cases. For some query predicates, this decision can be made using special properties of the query predicate, as we will see later in this thesis. In the general case, this decision can be made by choosing one representative world $w \in C_i$ (e.g. at random) from each class C_i, and evaluating the query predicate on this world. This yields at total run-time of $O(|\mathcal{C}|) \cdot O(\mathcal{I}(\phi, w))$, where $\mathcal{I}(\phi, w)$ is the time complexity of evaluating the query predicate ϕ on the certain database w. If this query predicate can be evaluated in polynomial time, i.e., if $O(\mathcal{I}(\phi, w)) \in O(poly(N))$, then the total run-time is in $O(poly(N))$. This is evident, since if $O(\mathcal{C})$ is in $O(poly(N))$, then $O(\mathcal{C}) \cdot O(\mathcal{I}(\phi, w))$ is in $O(poly(N)) \cdot O(poly(N))$ which is in $O(poly(N))$. For each class C_i, where the representative world satisfies ϕ, the corresponding probability $P(C_i)$ is added to the result probability.

The following lemma summarizes the assumptions that a query predicate has to satisfy in order to efficiently apply paradigm of finding equivalent worlds.

Lemma 4. *Given a query predicate ϕ and an uncertain database $\mathcal{DB}$ of size $N := |DB|$, we can answer ϕ on $\mathcal{DB}$ in polynomial time if the following four conditions are satisfied:*

I A traditional ψ query on non-uncertain data can be answered in polynomial time.

II we can identify a partitioning $\mathcal{C}$ of $\mathcal{W}$ into classes $C \in \mathcal{C}$ of equivalent worlds (see Definition 16.

III The number $|\mathcal{C}|$ of classes is at most polynomial in N.

IV The the total probability of a class $S \in \mathcal{C}$ can be computed in at most polynomial time.

Proof. Answering a ϕ query on $\mathcal{DB}$ requires to evaluate Equation 2.3 which we reformed into Equation 3.2 using property II. This requires to iterate over all $|\mathcal{C}|$ classes of equivalent worlds in polynomial time due to property III. For each class $C \in \mathcal{C}$, this requires to perform two tasks. The first task requires to compute the total probability of all worlds in C, and the second task requires to evaluate ϕ on a single possible world $w \in C$. The former task can be performed in polynomial time due to property IV. The later task requires to perform a crisp ϕ query on the (crisp) world w in polynomial time due to property I. □

3.3 Case Study: Sum of Independent Bernoulli Trials

In this chapter, the paradigm of equivalent worlds will be applied to efficiently solve the problem of computing the distribution of the sum of independent Bernoulli trials. An

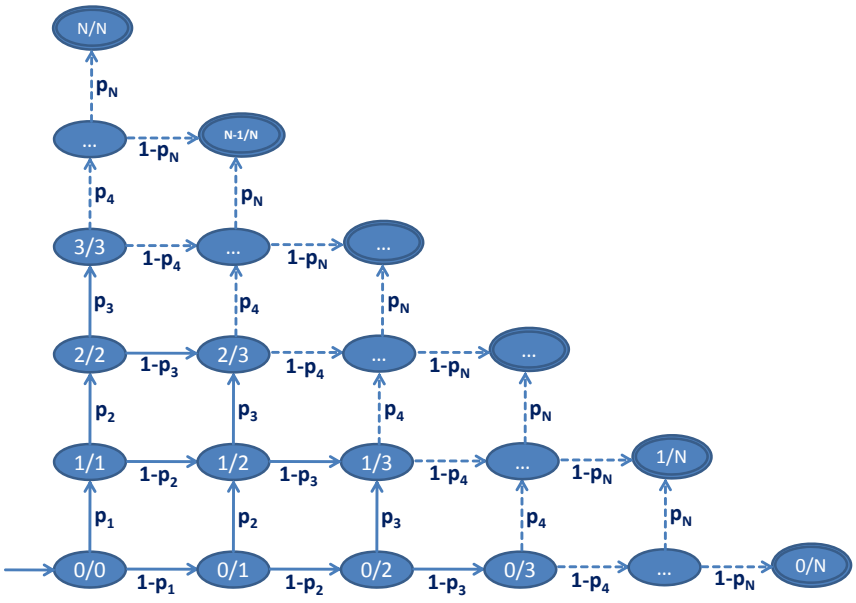


Figure 3.2: Deterministic finite automaton corresponding to the problem of the sum of independent Bernoulli trials.

efficient solution to this problem is the basis of many algorithms presented in this thesis. Then, in Part III, this paradigm of identifying worlds that are equivalent to the query predicate will be applied for the most relevant types of spatial queries on uncertain spatial data.

Let $X_1, ..., X_N$ be independent and not necessarily identically distributed Bernoulli trials, i.e., random variables that may only take values zero and one. Let $p_i := P(X_i = 1)$ denote the probability that random variable X_i has value one. In this section, we will show how to efficiently compute the distribution of the random variable

$$\sum_{i=1}^{N} X_i$$

without enumeration of all possible worlds. That is, for each $0 \leq k \leq N$, this section shows how to compute the probability $P(\sum_{i=1}^{N} X_i = k)$ that exactly k trials are successful.

In the following, two approaches to compute the probability distribution of $\sum_i X_i$ will be presented. While both approaches are equivalent in terms of time and space complexity, both approaches use significantly different techniques, which are applicable in different contexts, as we will see in the remainder of this thesis.

3.4 Poisson-Binomial Recurrence

The first approach iteratively computes the distribution of the sum of the first $1 \leq k \leq N$ Bernoulli variables given the distribution of the sum of the first $k-1$ Bernoulli variables.

To gain an intuition of how to do this efficiently, consider the deterministic finite automaton depicted in Figure 3.2.[2] The states (i/j) of this automaton correspond to the random event that out of the first j Bernoulli trials $X_1, ..., X_j$, exactly i trials have been successful. Initially, zero Bernoulli trials have been performed, out of which zero (trivially) were successful. This situation is represented by the initial state (0/0) in Figure 3.2. Evaluating the first Bernoulli trial X_1, there is two possible outcomes: The trial may be successful with a probability of p_1, leading to a state (1/1) where one out of one trials have been successful. Alternatively, the trial may be unsuccessful, with a probability of $1 - p_1$, leading to a state (0/1) where zero out of one trial have been successful. The second trial is then applied to both possible outcomes. If the first trial has not been successful, i.e., we are currently located in state (0/1), then there is again two outcomes for the second Bernoulli trial, leading to state (1/2) and (0/2) with a probability of p_2 and $1 - p_2$ respectively. If currently located in state (0/1), the two outcomes are state (2/2) and state (1/2) with the same probabilities. At this point, we have unified two different possible worlds that are equivalent with respect to $\sum_i X_i$: The world where trial one has been successful and trial two has not been successful, and the world where trial one has not been successful and trial two has been successful have been unified into state (1/2), representing both worlds. This unification was possible, since both paths leading to state (1/2) are equivalent with respect to the number of successful trials.

The three states (0/2), (1/2) and (2/2) are then subjected to the outcome of the third Bernoulli trial, leading to states (0/3), (1/3), (2/3) and (3/3). That is a total of four states for a total of $2^3 = 8$ possible worlds. In summary, the number of states in Figure 3.2 equals $\frac{N^2}{2}$. However, it is not yet clear how to compute the probability of a state (i/j) efficiently. Naively, we have to compute the sum over all paths leading to state (i/j). For example, the probability of state (2/3) is given by $p_1 \cdot p_2 \cdot (1 - p_3) + p_1 \cdot (1 - p_2) \cdot p_3 + (1 - p_1) \cdot p_2 \cdot p_3$. This naive computation requires to enumerate all $\binom{j}{i}$ combinations of paths leading to state (i/j).

For an efficient computation, we make the following observation: Each state of the DFA depicted in Figure 3.2 has at most two incoming edges. Thus, to compute the probability of a state (i/j), we only require the probabilities of states leading to (i/j). The states leading to state (i/j) are state $(i - 1/j - 1)$ and state $(i/j - 1)$. Given the probabilities $P(i - 1/j - 1)$ and $P(i/j - 1)$, we can compute the probability $P(i/j)$ of state (i/j) as follows:

$$P(i/j) = P(i - 1/j - 1) \cdot p_j + P(i, j - 1) \cdot (1 - p_j) \tag{3.3}$$

where

$$P(0/0) = 1 \text{ and } P(i/j) = 0 \text{ if } i > j \text{ or if } i < 0.$$

Equation 3.3 is known as the Poisson-Binomial Recurrence (To the best of our knowledge, the Poisson binomial recurrence was first introduced by [117]) and can be used to

[2]Note that this automaton is deterministic, despite the process of choosing a successor node being a random event. Once the Bernoulli trial corresponding to a node has been performed, the next node will be chosen deterministically, i.e., the upper node will be chosen if the trial was successful, and the right node will be chosen otherwise. Either way, there is exactly one successor node.

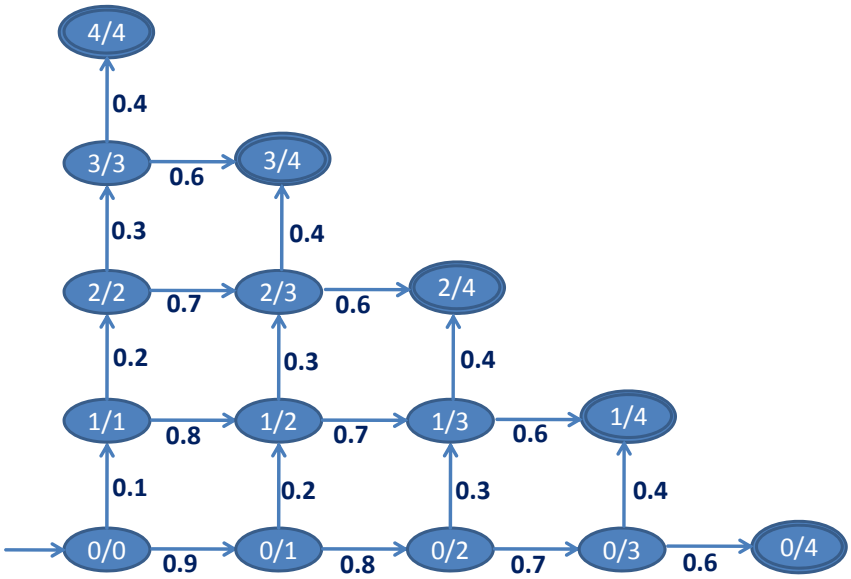


Figure 3.3: Example deterministic finite automaton for a total of four Bernoulli random variables.

compute the probabilities of states $(k/N), 0 \leq k \leq N$ which by definition, correspond to the probabilities $P(\sum_{i=1^N} X_i = k)$ that out of all N Bernoulli trials, exactly k trials are successful.

This approach follows the paradigm of equivalent worlds in each iteration k: The set of all 2^k possible worlds is partitioned into $k+1$ equivalent sets, each corresponding to a state i/k, where $i \leq k$. Each class contains only and all of the $\binom{k}{i}$ possible worlds where exactly i Bernoulli trails succeeded. The information about the particular sequence of the successful trials, i.e., which trials were successful and which were unsuccessful is lost. This information however, is no longer necessary to compute the distribution of $\sum_{i=0}^{N} X_i$, since for this random variable, we only need to know the number of successful trials, not their sequence. This abstraction allows to remove the combinatorial aspect of the problem.

An example showcasing the Poisson binomial recurrence is given in the following.

Example 14. *Let $N = 4$ and let $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$ and $p_4 = 0.4$. The corresponding DFA is depicted in Figure 3.3. The probability of state (0/0) is explicitly set to 1.0 in Equation 3.3. To compute the probability of state (0/1), we apply Equation 3.3 to compute*

$$P(0/1) = P(-1/0) \cdot p_1 + P(0/0) \cdot (1 - p_1).$$

with $P(-1/0) = 0$ and $P(0/0) = 1$ explicitly defined in Equation 3.3 this yields

$$P(0/1) = 0 \cdot p_1 + 1 \cdot (1 - p_1) = 0.9$$

Analogously, we obtain

$$P(1/1) = P(0/0) \cdot p_1 + P(1/0) \cdot (1 - p_1) = 1 \cdot p_1 = 0.1$$

Using these initial probabilities, we can continue to compute

$$P(0/2) = P(-1/1) \cdot p_2 + P(0/1) \cdot (1 - p_2) = 0 \cdot 0.2 + 0.9 \cdot 0.8 = 0.72$$

$$P(1/2) = P(0/1) \cdot p_2 + P(1/1) \cdot (1 - p_2) = 0.9 \cdot 0.2 + 0.1 \cdot 0.8 = 0.26$$

$$P(2/2) = P(1/1) \cdot p_2 + P(2/1) \cdot (1 - p_2) = 0.1 \cdot 0.2 + 0 \cdot 0.8 = 0.02$$

The probabilities $P(i/2), 0 \leq i \leq 2$ *can be used to compute*

$$P(0/3) = P(-1/2) \cdot p_3 + P(0/2) \cdot (1 - p_3) = 0 \cdot 0.3 + 0.72 \cdot 0.7 = 0.504$$

$$P(1/3) = P(0/2) \cdot p_3 + P(1/2) \cdot (1 - p_3) = 0.72 \cdot 0.3 + 0.26 \cdot 0.7 = 0.398$$

$$P(2/3) = P(1/2) \cdot p_3 + P(2/2) \cdot (1 - p_3) = 0.26 \cdot 0.3 + 0.02 \cdot 0.7 = 0.092$$

$$P(3/3) = P(2/2) \cdot p_3 + P(3/2) \cdot (1 - p_3) = 0.02 \cdot 0.3 + 0 \cdot 0.7 = 0.006$$

Finally, these probabilities can be used to derive the final distribution of the random variable $\sum_{i=1}^{4} X_i$*:*

$$P(0/4) = P(-1/3) \cdot p_4 + P(0/3) \cdot (1 - p_4) = 0 \cdot 0.4 + 0.504 \cdot 0.6 = 0.3024$$

$$P(1/4) = P(0/3) \cdot p_4 + P(1/3) \cdot (1 - p_4) = 0.504 \cdot 0.4 + 0.398 \cdot 0.6 = 0.4404$$

$$P(2/4) = P(1/3) \cdot p_4 + P(2/3) \cdot (1 - p_4) = 0.398 \cdot 0.4 + 0.092 \cdot 0.6 = 0.2144$$

$$P(3/4) = P(2/3) \cdot p_4 + P(3/3) \cdot (1 - p_4) = 0.092 \cdot 0.4 + 0.006 \cdot 0.6 = 0.0404$$

$$P(4/4) = P(3/3) \cdot p_4 + P(4/3) \cdot (1 - p_4) = 0.006 \cdot 0.4 + 0 \cdot 0.6 = 0.0024$$

These probabilities describe the PDF of $\sum_{i=1}^{4} X_i$ *by definition of* $P(i/j)$*.*

Complexity Analysis

To compute the distribution of $\sum_i X_i$ we require to compute each probability $P(i/j)$ for $0 \leq j \leq N, i \leq j$, yielding a total of $\frac{N^2}{2} \in O(N^2)$ probability computations. To compute any such probability, we have to evaluate Equation 3.3, which requires to look up four probabilities $P(i-1/j-1)$, $P(i/j-1)$, p_j and $1-p_j$, which can be performed in constant time. This yields a total runtime complexity of $O(N^2)$. The $O(N^2)$ space complexity required to store the matrix of probabilities $P(i/j)$ for $0 \leq j \leq N, i \leq j$ can be reduced to $O(N)$ by exploiting that in each iteration where the probabilities $P(i/k), 0 \leq i \leq k$ are computed, only the probabilities $P(i/k-1), 0 \leq i \leq k-1$ are required, and the result of previous iterations can be discarded. Thus, at most N probabilities have to be stored at a time.

3.5 Generating Functions

An alternative technique to compute the sum of independent Bernoulli variables is the generating functions technique. While showing the same complexity as the Poisson binomial recurrence, its advantage is its intuitiveness.

Represent each Bernoulli trial X_i by a polynomial $poly(X_i) = p_i \cdot x + (1 - p_i)$. Consider the generating function

$$\mathcal{F}^N = \prod_{i=1}^{N} poly(X_i) = \sum_{i=0}^{N} c_i x^i. \tag{3.4}$$

The coefficient c_i of x^i in the expansion of $\mathcal{F}^N$ equals the probability $P(\sum_{n=1}^{N} X_n = i)$ ([123]). For example, the monomial $0.25 \cdot x^4$ implies that with a probability of 0.25, the sum of all Bernoulli random variables equals four.

The expansion of N polynomials, each containing two monomials leads to a total of 2^N monomials, one monomial for each sequence of successful and unsuccessful Bernoulli trials, i.e., one monomial for each possible worlds. To reduce this complexity, again an iterative computation of $\mathcal{F}^N$, can be used, by exploiting that

$$\mathcal{F}^k = \mathcal{F}^{k-1} \cdot poly(X_k). \tag{3.5}$$

This rewriting of Equation 3.4 allows to inductively compute $\mathcal{F}^k$ from $\mathcal{F}^{k-1}$. The induction is started by computing the polynomial $\mathcal{F}^0$, which is the empty product which equals the neutral element of multiplication, i.e., $\mathcal{F}^0 = 1$. To understand the semantics of this polynomial, the polynomial $\mathcal{F}^0 = 1$ can be rewritten as $\mathcal{F}^0 = 1 \cdot x^0$, which we can interpret as the following tautology:"with a probability of one, the sum of all zero Bernoulli trials equals zero." After each iteration, we can unify monomials having the same exponent, leading to a total of at most $k + 1$ monomials after each iteration. This unification step allows to remove the combinatorial aspect of the problem, since any monomial x^i corresponds to a class of equivalent worlds, such that this class contains only and all of the worlds where the sum $\sum_{k=1}^{N} X_k = 1$. In each iteration, the number of these classes is k and the probability of each class is given by the coefficient of x^i.

An example showcasing the generating functions technique is given in the following. This examples uses the identical Bernoulli random variables used in Example 14.

Example 15. *Again, let $N = 4$ and let $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$ and $p_4 = 0.4$. We obtain the four generating polynomials $poly(X_1) = (0.1x + 0.9)$, $poly(X_2) = (0.2x + 0.8)$, $poly(X_3) = (0.3x + 0.7)$, and $poly(X_4) = (0.4x + 0.6)$. We trivially obtain $\mathcal{F}^0 = 1$. Using Equation 3.5 we get*

$$\mathcal{F}^1 = \mathcal{F}^0 \cdot poly(X_1) = 1 \cdot (0.1x + 0.9) = 0.1x + 0.9.$$

Semantically, this polynomial implies that out of the first one Bernoulli variables, the probability of having a sum of one is 0.1 (according to monomial $0.1x = 0.1x^1$, and the

probability of having a sum of zero is 0.9 *(according to monomial* $0.9 = 0.9x^0$*. Next, we compute* F^2*, again using Equation 3.5:*

$$\mathcal{F}^2 = \mathcal{F}^1 \cdot poly(X_2) = (0.1x^1 + 0.9x^0) \cdot (0.2x^1 + 0.8x^0) = 0.02x^1x^1 + 0.08x^1x^0 + 0.18x^0x^1 + 0.72x^0x^0$$

In this expansion, the monomials have deliberately not been unified to give an intuition of how the generating function techniques is able to identify and unify equivalent worlds. In the above expansion, there is one monomial for each possible world. For example, the monomial $0.18x^0x^1$ *represents the world where the first trial was unsuccessful (represented by the zero of the first exponent) and the second trial was succesful (represented by the one of the second exponent). The above notation allows to identify the sequence of successful and unsuccessful Bernouli trials, clearly leading to a total of* 2^k *possible worlds for* $\mathcal{F}^k$*. However, we know that we only need to compute the total number of successful trials, we do not need to know the sequence of successful trials. Thus, we need to treat worlds having the same number of successful Bernoulli trials equivalently, to avoid the enumeration of an exponential number of sequences. This is done implicitly by polynomial multiplication, exploiting that*

$$0.02x^1x^1 + 0.08x^1x^0 + 0.18x^0x^1 + 0.72x^0x^0 = 0.02x^2 + 0.08x^1 + 0.18x^1 + 0.72x^0$$

This representation no longer allows to distinguish the sequence of successful Bernouli trials. This loss of information is beneficial, as it allows to unify possible worlds having the same sum of Bernoulli trials.

$$0.02x^2 + 0.08x^1 + 0.18x^1 + 0.72x^0 = 0.02x^2 + 0.26x^1 + 0.72x^0$$

The remaining monomials represent equivalent class of possible worlds. For example, monomial $0.26x^1$ *represents all worlds having a total of one successful Bernoulli trials. This is evident, since the coefficient of this monomial was derived from the sum of both worlds having a total of one successful Bernoulli trials.In the next iteration, we compute:*

$$\mathcal{F}^3 = \mathcal{F}^2 \cdot poly(X_3) = (0.02x^2 + 0.26x^1 + 0.72x^0) \cdot (0.3x + 0.7)$$

$$= 0.006x^2x^1 + 0.014x^2x^0 + 0.078x^1x^1 + 0.182x^1x^0 + 0.216x^0x^1 + 0.504x^0x^0$$

This polynomial represents the three classes of possible worlds in $\mathcal{F}^2$ *combined with the two possible results of the third Bernoulli trial, yielding a total of* $3\dot{2}$ *monomials. Unification yields*

$$0.006x^2x^1 + 0.014x^2x^0 + 0.078x^1x^1 + 0.182x^1x^0 + 0.216x^0x^1 + 0.504x^0x^0 =$$

$$0.006x^3 + 0.092x^2 + 0.398x^1 + 0.504$$

The final generating function is given by

$$\mathcal{F}^4 = \mathcal{F}^3 \cdot poly(X_4)$$

$$= (0.006x^3 + 0.092x^2 + 0.398x^1 + 0.504) \cdot (0.4x + 0.6)$$

$$= 0.0024x^4 + 0.0036x^3 + 0.0368x^3 + 0.0552x^2 + 0.1592x^2 + 0.2388x^1 + 0.2016x^1 + 0.3024x^0$$

$$= 0.0024x^4 + 0.0404x^3 + 0.2144x^2 + 0.4404x + 0.3024$$

This polynomial describes the PDF of $\sum_{i=1}^{4} X_i$, since each monomial $c_i x^i$ implies that the probability, that out of all four Bernoulli trials, the total number of successful events equals i, is c_i. Thus, we get $P(\sum_{i=1}^{4} X_i = 0) = 0.0024$, $P(\sum_{i=1}^{4} X_i = 1) = 0.0404$, $P(\sum_{i=1}^{4} X_i = 2) = 0.2144$, $P(\sum_{i=1}^{4} X_i = 3) = 0.4404$ and $P(\sum_{i=1}^{4} X_i = 4) = 0.3024$. Note that this result equals the result we obtained by using the Poisson binomial recurrence in the previous section.

Complexity Analysis

The generating function technique requires a total of N iterations. In each iteration $1 \leq k \leq N$, a polynomial of degree k, and thus of maximum length $k+1$, is multiplied with a polynomial of degree 1, thus having a length of 2. This requires to compute a total of $(k+1) \cdot 2)$ monomials in each iteration, each requiring a scalar multiplication. Thus leads to a total time complexity of $\sum_{i=1}^{N} 2k+2 \in O(N^2)$ for the polynomial expansions. Unification of a polynomial of length k can be done in $O(k)$ time, exploiting that the polynomials are sorted by the exponent after expansion. Unification at each iteration leads to a $O(n^2)$ complexity for the unification step. This results in a total complexity of $O(n^2)$, similar to the Poisson binomial recurrence approach.

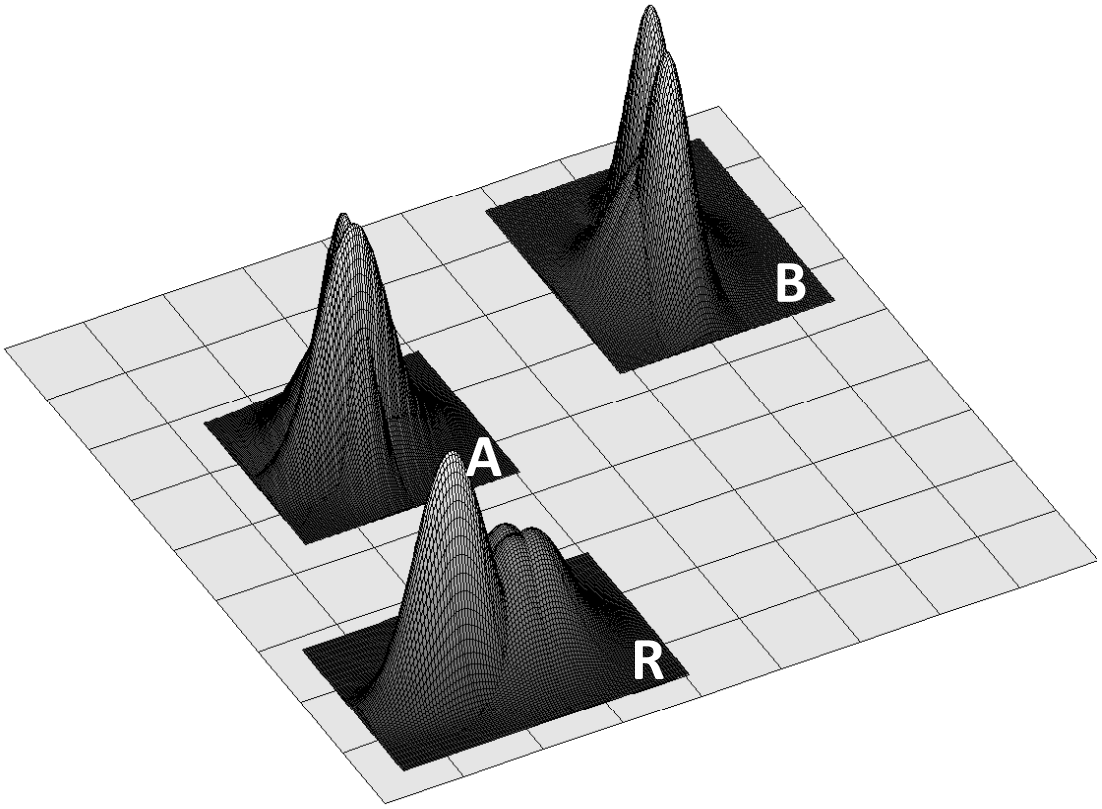
An advantage of the generating function approach is that this naive polynomial multiplication can be accelerated using Discrete Fourier Transform (DFT). This technique allows to reduce to total complexity of computing the sum of N Bernoulli random variables to $O(Nlog^2N)$([126]). This acceleration is achieved by exploiting that DFT allows to expand two polynomials of size k in $O(klogk)$ time. Equi-sized polynomials are obtained in the approach of [126], by using a divide and conquer approach, that iteratively divides the set of N Bernoulli trials into two equi-sized sets. Their recursive algorithm then combines these results by performing a polynomial multiplication of the generating polynomials of each set. More details of this algorithm can be found in [126].

3.6 Summary

Both presented techniques, the Poisson-Binomial recurrence and the generating functions technique allow to efficiently compute the distribution of a Poisson-Binomial distributed random variable, i.e., the sum of independent but not necessarily identical distributed Bernoulli trials. Both approaches achieve this efficient computation, by unifying sets of possible worlds, and treating the resulting set as a whole, rather than treating each world individually. In particular, the Poisson-Binomial recurrence unifies, in each iteration, all world having the same number of successful Bernoulli trials out of the currently considered Bernoulli trials. The generating functions techniques unifies worlds using simple algebra, adding monomials having the same exponent. Thus, both approaches identify and unify sets of possible worlds, thus following the paradigm of equivalent worlds introduced in this chapter. The efficient computation resulting from either technique is paramount in answering many probabilistic queries on spatial and spatio-temporal data. Both techniques will be applied, adapted and improved throughout this thesis.

Part III

Probabilistic Spatial Queries on Uncertain Data



In this section, the paradigm of equivalent worlds presented in Chapter 3 is applied to permit efficient spatial similarity search on uncertain spatial data. For the most relevant spatial query types, which have been presented in Chapter 1, efficient solutions are presented. This part is structured as follows.

- Chapter 4 will give an efficient solution to answer range queries on uncertain data, for both cases where the query object is a (certain) point, and the case where the query object is uncertain itself. Furthermore, the problem of answering range queries on uncertain data will be extended by considering the range count query, which is to return the number of objects within a given range of a query object. In an uncertain database, this number is a random variable, for which this section will give an efficient solution to compute its distribution. The solutions shown in this chapter are rather straight-forward, given the paradigm of equivalent worlds. Nonetheless, solutions for range queries on uncertain data are necessary for completeness of this thesis, and at the same time, show-case the function of the paradigm of equivalent worlds. Some of the results of this chapter have been published in [30].

- Chapter 5 presents a novel a approach to facilitate spatial pruning of uncertain objects that are conservatively approximated by minimal bounding boxes. This spatial pruning approach is a key technique to boost efficiency of similarity queries such as k-nearest neighbor queries, ranking queries and reverse k-nearest neighbor queries on uncertain data. Parts of this chapter have been published at the ACM SIGMOD Conference in 2010 ([65]) as a full paper.

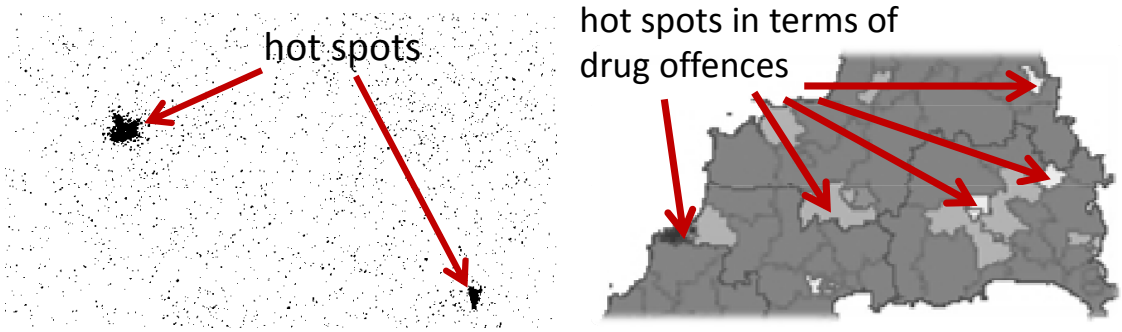
- In Chapter 6 a solution for kNN-queries on uncertain data is presented. Here, the main challenge is that the predicate of an object being a kNN of a query object is a random variable, which stochastically depends on the position of other objects. An efficient solution to handle these dependencies is given in this section. Parts of this chapter have been published at the IEEE International Conference on Data Engineering (ICDE) 2011 ([22]) as a full paper.

- Chapter 7 extends chapter 6 by giving a solution to the problem of answering spatial ranking queries in uncertain databases. This section will show how to efficiently compute the rank distribution, i.e., the probability of each rank of each object in an uncertain database. Parts of this chapter have been published at the IEEE Transactions on Knowledge and Data Engineering journal 2010 ([25]) as a regular paper.

- Finally, Chapter 8 will give the first efficient solution for the problem of reverse kNN queries in uncertain databases. Parts of this chapter have been published in the Proceedings of the VLDB Endowment (PVLDB), Volume 4, 2011 ([23]) as a full paper.

Chapter 4

Probabilistic Range Queries on Uncertain Data



(a) Astrological hot items in terms of interesting constellations.

(b) Hot item detection for crime defence applications.

Figure 4.1: Applications for hot item detection.

4.1 Introduction

The detection of dense regions in a feature space is paramount in a variety of several density-based data mining techniques, in particular density-based clustering [68, 161], density-based outlier detection [38] and other density-based mining applications [113, 175]. We call a region R, for which a sufficiently large population of objects in $\mathcal{DB}$ exists a *dense region* or *hot location*. Analogously, we call an object o, for which a sufficiently large number of other objects in $\mathcal{DB}$ are similar to o, a *hot item*. Intuitively, an item that shares its attributes with a lot of other items could be potentially interesting as its shows a typical occurrence of items in the database. Deciding for a given item, whether it is an hot item is an important subtask in density-based clustering algorithms such as DBSCAN [68], where such items are called core items or core points. Further application areas where the detection of *hot items* is potentially important exemplarily include scientific applications, e.g. astrophysics (cf. Figure 4.1(a) showing a clipped capture of a star field

in the Sagittarius area), biomedical, sociological and economic applications. In particular, the following applications give a good motivation for the efficient detection of hot items:
Pre-detection of criminal activities: After a soccer game one might be interested in the detection of larger groups of hooligans that should be accompanied by guards in order to avoid criminal excess. If we assume that the locations of all hooligans are monitored, then it would be interesting which of these individuals have a lot of other hooligans in their immediate vicinity. Another example is the detection of outstanding crime delicts, e.g. cases of drug abuse in areas with high population of drug offences as depicted in Figure 4.1(b).[1] To find hot items, the density of other items in the vicinity of the probed object has to be assessed. In traditional databases, an ϵ range query can be utilized to perform this task: if the number of other database objects inside a certain range ϵ around an object exceeds a minimal threshold $minItems$, this object is declared a hot item, formally

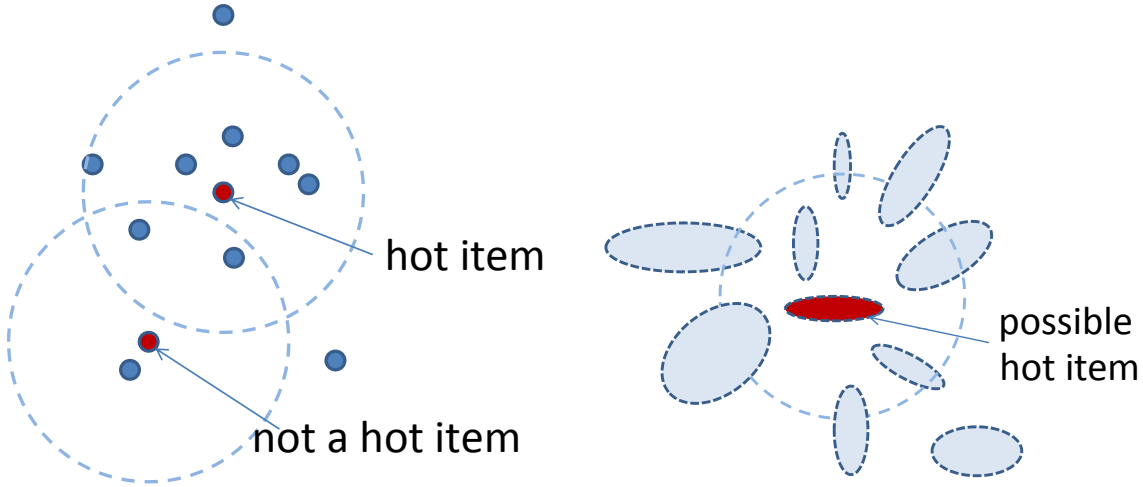
Definition 17 (Hot Item)**.** *Given a spatial point database $\mathcal{DB}$, a scalar spatial distance threshold ϵ and an integer minimum population threshold $minItems$. An object $o \in \mathcal{DB}$ is called* hot item*, if and only if*

$$|\{o' \in \mathcal{DB} \backslash \{o\} : dist(o, o') < \epsilon\}| > minItems.$$

Both parameters ϵ and $minItems$ are user-specified and application dependent. An example is depicted in Figure 4.2(a), where the ϵ-range for two objects of a exemplary database ϵ-range is depicted. For a parameter value of and $minItems = 5$, only one of these items is a hot item. On uncertain data, the task of deciding whether a database object is a hot item becomes more challenging as seen in Figure 4.2(b). For the highlighted object o, the predicate $dist(o, o') < \epsilon$ that o is sufficiently close to another object o' becomes a random variable, that may be true with some probability, and false otherwise. Consequently, the number of objects in range of o also turns into a random variable having a probability mass function defined on $\mathbb{N}_0$.

This chapter will show how to efficiently answer range queries on uncertain data. Related work is presented in the following Section 4.2. Next, the paradigm of equivalent worlds (c.f. Chapter 3) is applied in Section 4.3 in order to find an efficient solution for the problem of answering probabilistic range queries on uncertain data in the special case where the query object is a (certain) point. While this type of query is rather trivial to answer efficiently due to object independence in the X-tuple model, it is a good showcase of the paradigm of equivalent worlds introduced in Chapter 3. This solution is generalized to the case of an uncertain query point in Section 4.4 by again using the paradigm of equivalent worlds. Since range queries do not directly return a scalar measuring the density of a region, but rather return a set of potential result objects associated with their respective probability to be a result, Section 4.5 presents an efficient solution to the problem of computing the distribution of the total number of objects located in a given range, which allows to assess the probability that an object is hot. This problem is particular challenging, since distances between objects are stochastically dependent even in the case

[1] source: www.amethyst.gov.uk/crime_map/crimedrugs.htm



(a) Hot items in certain data.

(b) Hot items in uncertain data.

Figure 4.2: Examples of hot items.

where the location of objects are stochastically independent. An detailed explanation and an example of this problem is also given in Section 4.5. Again, an efficient solution can again be found using the paradigm of equivalent worlds. To allow an experimental evaluation of the efficiency of this approach, a straightforward approach as well as a competitive approach is introduced. The results of an experimental evaluation of all approaches are presented in Section 4.6 before this chapter is concluded in Section 4.7.

4.2 Related Work

The aspect to identify objects that are similar to a given amount of other objects is the basis of several density-based algorithms for discovering clusters and outliers. There exist approaches for density-based clustering of uncertain data, e.g. [111] which are quite related to our approach. However the proposed model used to determine the probabilistic density does not respect the mutual exclusiveness of alternative attribute values. The missing conditional probability in their approach leads to approximative results only which disqualifies this approach from the accurate detection of hot items. A more detailed description of this problem is given in Section 4.5.

This problem does not occur in the special case where the query object is given by a single (certain) point rather than an uncertain object. For this case, and efficient solution for range search on uncertain data has been proposed in [184]. This approach utilized an index structure, called the U-tree for minimizing the query overhead. This index structure employs so called *probabilistic constrained regions*, which essentially approximate the marginal distribution of uncertain objects, to permit probabilistic pruning.

The detection of core objects can be efficiently supported by a similarity join query used in a preprocessing step, in particular the distance range self-join. Approaches for an efficient join on uncertain data are proposed in [109]. The main advantage of this approach is that sampled positions in space can efficiently be indexed using traditional spatial access methods thus allowing to reduce the computational complexity of complex query types. Our approach exploits the similarity join approach proposed in [109]. However, the cost of the probabilistic detection of hot items are originally highly CPU-bound which is demonstrated in the experimental evaluation (cf. Section 4.6).

4.3 Probabilistic Range Queries on Uncertain Data: Certain Query

Adhering to possible world semantics and object based answer semantics, a spatial range query for a (certain) query point q on uncertain data is defined as follows:

Definition 18 (Probabilistic Range Query)**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let q be a query point, and let ϵ be a positive real value. A spatial range query computes, for each database object, the probability of having a distance less than ϵ to q.*

$$\epsilon\text{-}range(q, \mathcal{DB}) := \{(U \in \mathcal{DB}, P(dist(q, U) \leq \epsilon)\}$$

The main challenge of answer a probabilistic range query is to compute the probability $P(dist(q, U) \leq \epsilon))$ of the predicate $dist(q, U) \leq \epsilon)$ for each $U \in \mathcal{DB}$. Naively, we can use possible worlds semantics to compute this probability by using Equation 2.3 to obtain

$$P(dist(q, U) \leq \epsilon)) = \sum_{w \in \mathcal{W}} \mathcal{I}(dist(q, U) \leq \epsilon, w) \cdot P(w)$$

Clearly, this solution is inefficient, as it requires to enumerate all possible possible. Towards an efficient solution, we make the following observation.

Corollary 2. *Let u be an instance of uncertain object U. The set of possible worlds where $U = u$ is a class of worlds equivalent with respect to the predicate $dist(q, U) \leq \epsilon$.*

Proof. Corollary 2 follows trivially from the fact that the distance $dist(q, U)$ is independent from all database objects $U' \in \mathcal{DB} \setminus \{U\}$. □

A partitioning of $\mathcal{W}$ can be derived trivially using Corollary 2.

Corollary 3. *Let $\{u_1, ..., u_m\}$ the set of possible instances of uncertain object U, where m is the number of alternatives of U. Let $S_i \subseteq \mathcal{W}$ be the set of possible worlds where $U = u_i$. Furthermore, let $S_0 \subseteq \mathcal{W}$ be the set of possible worlds where U does not exist.*[2] *The set $\{S_0, ..., S_m\}$, is a disjunctives partitioning of $\mathcal{W}$.*

Proof. Corollary 3 follows from the property of mutual exclusive alternatives in the x-tuple model, where in each world, every object (in particular object U) has at most one alternative value. □

Corollary 3 leads to the following Lemma:

Lemma 5. *The probability $P(dist(q, U) \leq \epsilon))$ can be computed in polynomial time.*

Proof. Reconsider the paradigm of equivalent worlds presented in Chapter 3. Property I is satisfied trivially, as a range query on certain data can be answered in linear time via a single database scan. Corollary 3 directly satisfies property II. The number of equivalent classes is equal to one plus the number of alternatives of object U, which is assumed to be finite and constant in the size of the database in the x-tuple model. Thus property III is satisfied. Finally, the probability $P(S_i)$ of class S_i is given by the x-tuple model due to the assumption of independence objects: The probability of S_i equals the probability $P(u_i)$ of instance u_i for $1 \leq i \leq m$ and equals the probability $1 - \sum_{i=1}^{m} P(u_i)$ for $i = 0$.

With all four properties being satisfied, we can apply Equation 3.2:

$$P(\phi(\mathcal{DB})) = \sum_{S \in \mathcal{S}} \mathcal{I}(\phi, w \in S) \cdot \sum_{w \in S} P(w).$$

Substitution of the abstract predicate $\phi(\mathcal{DB}) = dist(q, U) \leq \epsilon)$ and substitution of the abstract partition $\mathcal{S} = \{S_0, ..., S_m\}$ yields

$$P(dist(q, U) \leq \epsilon)) = \sum_{S_i \in \mathcal{S}} \mathcal{I}(dist(q, U) \leq \epsilon, w \in S_i) \cdot \sum_{w \in S_i} P(w).$$

Substitution of the total probability $P(S_i) = \sum_{w \in S_i} P(w) = P(u_i)$ of each class yields

$$P(dist(q, U) \leq \epsilon)) = \sum_{S_i \in \mathcal{S}} \mathcal{I}(dist(q, U) \leq \epsilon, w \in S_i) \cdot P(u_i), \tag{4.1}$$

[2] This set may only be non-empty if existential uncertainty is allowed.

where

$$\mathcal{I}(dist(q, U) \leq \epsilon, w \in S_i)$$

is an indicator function that returns one if $dist(q, o) \leq \epsilon$ in world w and zero otherwise. We can exploit that for each world in class S_i, it holds that $U = u_i$ by definition of S_i. thus we can instead use the indicator function

$$\mathcal{I}(dist(q, u_i) \leq \epsilon)$$

that returns one if $dist(q, u_i) \leq \epsilon$ and zero otherwise. □

In summary, Equation 4.1 can be computed efficiently, since

- the indicator function $\mathcal{I}(dist(q, u_i) \leq \epsilon)$ can be evaluated efficiently, by performing a single distance computation on certain data,
- and the probability $P(u_i)$ of a single instance of an uncertain object U can be evaluated efficiently, by evaluating the probability mass function of U, which is given by the uncertain data model.

While answering range queries efficiently is a rather trivial result, the main role of this section was to give an example of how the paradigm of finding equivalent worlds can be applied to a simple query predicate. The remainder of this chapter, as well subsequent chapters of this thesis will present more challenging spatial query types.

4.4 Probabilistic Range Queries on Uncertain Data: Uncertain Query

Again, adhering to possible world semantics and object based answer semantics, a spatial range query for an uncertain query point q on uncertain data is defined as follows:

Definition 19 (Probabilistic Range Query)**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let Q be an uncertain query object, and let ϵ be a positive real value. A spatial range query computes, for each database object, the probability of having a distance less than ϵ to Q.*

$$\epsilon - range(Q, \mathcal{DB}) := \{(U \in \mathcal{DB}, P(dist(Q, U) \leq \epsilon)\}$$

Definition 19 is almost equal to Definition 18, except that the query object is no longer assumed to be a certain point, but rather may itself be an uncertain object.

To compute the probability $P(dist(Q, U) \leq \epsilon)$, we first need to formally define the distance between two uncertain objects $dist(Q, U)$. Clearly, for two objects having uncertain locations, the distance between these two objects is also uncertain, i.e., a random variable. In the discrete case, a probabilistic distance function is defined as follows.

Definition 20 (Probabilistic Distance)**.** *Let $\mathcal{DB}$ be an uncertain spatial databases and let $dist : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_0^+$ be a distance function on (certain) points. Furthermore, let U_i and U_j be two uncertain objects. A probabilistic distance $dist(U_i, U_j)$ returns a cumulative probability density function (CDF) of the distance between U_i and U_j.*

$$dist : \mathcal{DB} \times \mathcal{DB} \to (\mathbb{R}_0 \to [0, 1])$$

$$dist(U_i, U_j) = \{(d \in \mathbb{R}_0, P(dist(U_i, U_j) \leq d) \in [0, 1])\},$$

where

$$P(dist(U_i, U_j) \leq d) = \sum_{w \in \mathcal{W}} \mathcal{I}(dist(U_i, U_j) \leq d, w) \cdot P(w).$$

Note that the function $dist(\cdot, \cdot)$ has previously been defined between two certain points. This function is overloaded deliberately. It should be clear from the context whether the traditional distance $dist : \mathbb{R}^d \times \mathbb{R}^d$ defined on (certain) points, or the uncertain version $dist : \mathcal{DB} \times \mathcal{DB} \to (\mathbb{R}_0 \to [0, 1])$ defined on uncertain objects is used.

Lemma 6. *The probability $P(dist(U_i, U_j) \leq d)$ can be computed in polynomial time.*

Proof. To compute the probability $P(dist(U_i, U_j) \leq d)$ we observe that exploiting object independence, the probability $P(dist(U_i, U_j) \leq d)$ depends only on the positions of uncertain objects U_i and U_j, and is independent of any other database object in $\mathcal{DB} \setminus \{U_i, U_j\}$. This observation allows to easily find sets of possible worlds that are equivalent with respect to the random event $dist(U_i, U_j) \leq d$: Let $x \in U_i$ be a possible location of U_i, and let $y \in U_i$ be a possible location of U_j, then any world $w \in C_{x,y} := \{w \in \mathcal{W} | U_i = x, U_j = y\}$ is equivalent with respect to the random event $dist(U_i, U_j) \leq d$. Thus the equivalence $\forall w_1, w_2 \in C_{x,y} : dist(w_1.U_i, w_1.U_j) \leq d \Leftrightarrow \forall dist(w_2.U_i, w_2.U_j) \leq d$ holds. Formally, this equivalence is evident, by substitution of $w_1.U_i = w_2.U_i = x$ and $w_1.U_j = w_2.U_j = y$. Since there exists one equivalent class $C_{x,y}$ for each $x \in U_i$ and each $y \in U_j$, the number of equivalent classes equals $|U_i| \cdot |U_j|$, where $|U_i|$ ($|U_j|$) is the number of possible locations of U_i (U_j). Thus, conditions II and III of Lemma 4 are satisfied. Condition I is satisfied trivially, assuming that the distance function $dist(x, y)$ for two (certain) points x and y can be computed in polynomial time, which is the case for Euclidean distance. Finally, condition IV requires to compute the total probability of a equivalent class to be computed efficiently, i.e., the probability

$$P(C_{x,y}) = \sum_{w \in C_{x,y}} P(w)$$

has to be computed efficiently. By definition of $C_{x,y}$, this equation can be rewritten as

$$\sum_{w \in C_{x,y}} P(w) = \sum_{\{w \in \mathcal{W} | U_i = x, U_j = y\}} P(w).$$

The right-hand side of above equation aggregates the probabilities of all worlds where $U_i = x, U_j = y$. Using the indicator function $\mathcal{I}(U_i = x \wedge U_j = y, w)$ that returns 1 if the predicate $U_i = x \vee U_j = y$ holds in world w, this can be rewritten as

$$\sum_{\{w \in \mathcal{W} | U_i = x, U_j = y\}} P(w) = \sum_{\{w \in \mathcal{W}\}} \mathcal{I}(U_i = x \wedge U_j = y, w) P(w).$$

Using the definition of possible world semantics (Equation 2.3), we obtain

$$\sum_{\{w \in \mathcal{W}\}} \mathcal{I}(U_i = x \wedge U_j = y, w) P(w) = P(U_i = x \wedge U_j = y)$$

Exploiting independence between U_i and U_j we finally obtain

$$P(U_i = x \wedge U_j = y) = P(U_i = x) \cdot P(U_j = y),$$

which can be computed in constant time by looking up the probabilities $P(U_i = x)$ and $P(U_j = y)$ given by the models of U_i and U_j.

Thus, condition *IV* holds, and Lemma 4 is applicable to compute $P(dist(U_i, U_j) \leq d)$ efficiently by

$$P(dist(U_i, U_j) \leq d) = \sum_{C_{x,y}, x \in U_i, y \in U_y} \mathcal{I}(dist(w.U_i, w.U_j) \leq d, w) P(C_{x,y}) \tag{4.2}$$

This equation requires to iterate over all equivalent classes $C_{x,y}, x \in U_i, y \in U_y$, summing up the probabilities $P(C_{x,y})$ for each class where $dist(w.U_i, w.U_j) \leq d$ for a world $w \in C_{x,y}$. Exploiting that the probability $P(C_{x,y})$ of each class $C_{x,y}$ can be computed in constant time, the total time complexity of computing $P(dist(U_i, U_j) \leq d)$ is in $O(|C|)$ where $|C|$ is the number of classes, which equals $O(|x| \cdot |y|)$ where $|x|$ and $|y|$ denote the number of possible locations of objects x and y. □

Two answer a probabilistic ϵ-range query as defined in Definition 18, we can apply Equation 4.2 to compute the probabilities $P(dist(Q, o) \leq \epsilon)$ for each $o \in \mathcal{DB}$ by substituting d by ϵ, U_i by Q, and U_j by o for each $o \in \mathcal{DB}$. This yields a total run-time of $O(\sum_{o \in \mathcal{DB}} |Q| \cdot |o|)$ which is in $O(|\mathcal{DB}| \cdot |Q| \cdot max_{o \in \mathcal{DB}} |o|)$.

4.5 Range Count Queries on Uncertain Data

This section gives an efficient solution of the problem of finding the distribution of the total number $|\{U \in \mathcal{DB} | dist(Q, U) \leq \epsilon)\}|$ of database objects located within an ϵ-range around a query object Q. At first glance, this task may seem rather straightforward, given the probabilities $P(dist(Q, U) \leq \epsilon)$ for each $U \in \mathcal{DB}$, which can be computed efficiently as shown in Section 4.4. Given these probabilities of each object to be located in ϵ-range of Q, it may seem possible to use the techniques presented in Section 3.3 to compute the sum of independent Bernoulli trials. However, the events $dist(Q, U_1 \in \mathcal{DB}) < \epsilon$ and $dist(Q, U_2 \in \mathcal{DB}) < \epsilon$ are not stochastically independent, as both events depends on the position of Q. This problem described in detail in the following example.

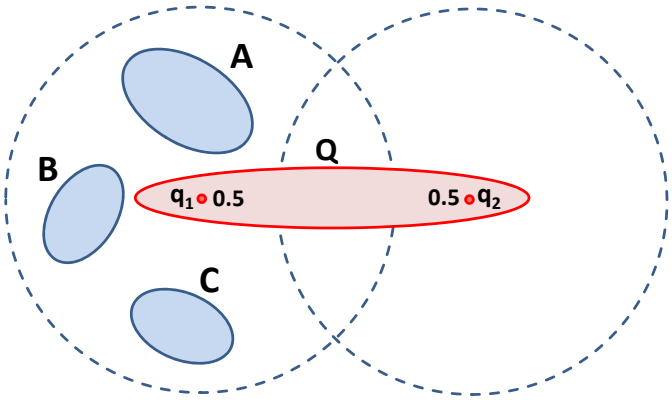


Figure 4.3: An example database showing the stochastic dependencies between probabilistic distances.

Example 16. *To illustrate this problem, consider Figure 4.3. It shows a query object Q having two alternative positions having a probability of* 0.5 *each. If Q has position q_1, i.e., assuming that $Q = q_1$, we see that all three database objects A, B and C are certainly located in the ϵ-range of q_1, which is depicted by the large circle centered at q_1. If Q has position q_2, that is if $Q = q_2$, then all objects will certainly be located outside of the ϵ-range of q_2. Thus, it is clear that for any object $O \in \{A, B, C\}$ it holds that $P(dist(Q, O) \leq \epsilon) = 0.5$: With a probability of* 0.5*, Q is at location q_1 and certainly has O in it's range, and with a probability of* 0.5*, Q is at location q_2 and O is certainly out of range of Q. If we (incorrectly) assume independence between the events $dist(Q, O) \leq \epsilon$, we can compute the distribution of the number $count(Q, \epsilon, \mathcal{DB}) := |\{O \in \mathcal{DB}|\}|dist(Q, O) \leq \epsilon)$ of objects in range of Q using the technique of generating functions by*

$$\mathcal{F}(x) = (0.5 + 0.5x) \cdot (0.5 + 0.5x) \cdot (0.5 + 0.5x) = 0.125x^3 + 0.375x^2 + 0.375x + 0.125.$$

Interpreting the semantics of the monomials of this expansion yields the following (incorrect) probabilities $P(count(Q, \epsilon, \mathcal{DB}) = 0) = P(count(Q, \epsilon, \mathcal{DB}) = 3) = 0.125$ and $P(count(Q, \epsilon, \mathcal{DB}) = 1) = P(count(Q, \epsilon, \mathcal{DB}) = 2) = 0.375$. We can clearly see that these probabilities must be wrong, since in all worlds where $Q = q_1$, we know that $count(Q, \epsilon, \mathcal{DB})$ must equal 3. Knowing that $P(Q = q_1)$ we get that $P(count(Q, \epsilon, \mathcal{DB}) = 3) = 0.5$. Analogously, for the case $Q = q_2$ we get $P(count(Q, \epsilon, \mathcal{DB}) = 0) = 0.5$. The wrong results returned by the above generating functions technique are a result of the incorrect assumption of independence between the three stochastic events $dist(Q, O \in \{A, B, C\}) \leq \epsilon$. In this example, these events are indeed highly correlated, since it holds that $dist(Q, A) \leq \epsilon$ if and only if $dist(Q, A) \leq \epsilon$ if and only if $dist(Q, A) \leq \epsilon$. Thus, the three events are equivalent, thus having a correlation of one.

4.5.1 Probabilistic Hot Items

The problem of computing the density in the range of a query object is defined as follows

Definition 21 (Probabilistic Hot Item)**.** *Let $\mathcal{DB}$ be an uncertain spatial database, let ϵ be a real value and let minItems be a positive integer. A database object $U \in \mathcal{DB}$ is called a hot item if*

$$U \text{ is a } \textbf{\textit{hot item}} := |\{U' \in DB \setminus \{U\} | dist(U, U') \leq \epsilon\}| > minItems$$

Using possible worlds semantics, the probability of this random event is given by

$$P(|\{U' \in DB | dist(U, U') \leq \epsilon\}| > minItems) = \sum_{w \in \mathcal{W}} \mathcal{I}(\{U' \in DB \setminus \{U\} | dist(U, U') \leq \epsilon\}, \mathcal{DB}) \cdot P(w) \tag{4.3}$$

Based on the uncertainty models and the corresponding definitions given above, we can compute hot items in uncertain data in a probabilistic way. However, we have to solve the problem of distance-dependencies of the uncertain attributes. Though we assume that the locations of uncertain spatial objects are independent of each other, we have to respect that the random variables $dist(U_1, U_2)$ and $dist(U_3, U_4)$ for objects $U_1, ..., U_4 \in \mathcal{DB}$ are mutually dependent if $\{U_1, U_2\} \cap \{U_3, U_4\} \neq \emptyset$. Obviously, the problem here is that the uncertain object Q is used in both random variables $P(dist(Q, A) \leq \epsilon)$ and $P(dist(Q, B) \leq \epsilon)$, rendering these variables mutually dependent, despite the independence between objects A, B and Q, as shown in Example 16. To avoid this dependency, we perform a partitioning of possible worlds into subsets of worlds: One partition for each possible position of $q \in Q$. In each such partition, the random variables $P(dist(q, A) \leq \epsilon)$ and $P(dist(q, B) \leq \epsilon)$ are independent, since the only involved uncertain objects are A and B, which are independent by model definition.

Definition 22 (Conditional Probabilistic Hot Item)**.** *Given a database $\mathcal{DB}$ with uncertain objects and a minimum population threshold minItems. Furthermore, let $\varepsilon \in \mathbb{R}_0^+$ be a scalar. Under the condition that an uncertain object $U \in \mathcal{DB}$ has a certain location $x \in \mathbb{R}^d$, the probability that U is a* hot item*, given that $U = x$ is denoted as*

$$P(U \text{ is a } \textbf{\textit{hot item}} | U = x)$$

This definition allows to compute the probability that U is a hot item as follows:

Corollary 4.

$$P(U \text{ is a } \textbf{\textit{hot item}}) = \sum_{x \in U} P(U \text{ is a } \textbf{\textit{hot item}} | U = x) \cdot P(x)$$

Proof. Corollary 4 follows directly from the law of total probability ([220]). □

Corollary 4 allows to reduce the problem of computing the probability that an uncertain item is hot, to the problem of computing the probability that a single location is hot. To compute the probability

$$P(U \text{ is a } \textbf{hot item} | U = x) = P(|\{o' \in DB \setminus \{U\} | dist(U, U') \leq \epsilon\}| > minItems | U = x)$$

$$= P(|\{U' \in DB \setminus \{U\} | dist(x, U') \leq \epsilon\}| > minItems)$$

we first start by computing the probabilities

$$P(dist(x, U') \leq \epsilon)$$

using the technique presented in Section 4.3. Note that, depending on the value of ϵ, usually only a small portion $\mathcal{DB}' \subset \mathcal{DB}$ of the database has a non-zero probability to be in ϵ-range of x. A quick search of those objects which have to be taken into account can be efficiently supported by means of an index structure, e.g. the R*-tree. In particular, the index supported ε-range join [39] can be used to speed-up the search as proposed in [34]. Here, approximative representations like the minimal bounding rectangle (mbr) of an uncertain object are very appropriate to be used as index key for a filter step following the multi-step query processing paradigm. A solution for the ε-range join on uncertain data is proposed in [109] which can be used as a preprocessing step for our proposed algorithm for the detection of hot items.

To efficiently compute the distribution of the number $|\{U' \in \mathcal{DB} \setminus \{U\} | dist(x, O') \leq \epsilon\}|$ of objects close to instance $x \in U$, we proceed as follows. Since each random event $dist(x, U_i \in \mathcal{DB} \setminus \{U\}) \leq \epsilon$ follows a Binomial distribution, having two possible values true and false, we can simply define a Bernoulli distributed random variable B_i that returns one if $dist(x, U_i) \leq \epsilon$ and zero otherwise. By definition, the sum

$$\sum_{U_i \in \mathcal{DB} \setminus \{U\}} B_i$$

corresponds to the number of random events $dist(x, U_i \in \mathcal{DB}) \leq \epsilon$ that hold true. These events $dist(x, U_i) \leq \epsilon$ and $dist(x, U_j) \leq \epsilon, U_i, U_j \in \mathcal{DB} \setminus \{U\}, U_i \neq U_j$ are now independently distributed, since x is now a fixed location and not a random variable. Consequently, the corresponding Beroulli random variable B_i and B_j are also mutually independent. Thus, we can use the techniques presented in Section 3.3 to compute the distribution of $\sum_{U_i \in \mathcal{DB} \setminus \{U\}} B_i$ efficiently. This distribution can be used to compute $P(U \text{ is a } \textbf{hot item} | U = x)$ for each possible alternative $x \in U$ of U. Given these probabilities, we can apply Corollary 4 to compute the probability $P(U \text{ is a } \textbf{hot item})$.

4.6 Experimental Evaluation

In Section 3.3 and in Section 4.5.1 the paradigm of equivalent worlds has been used to efficiently compute the probabilistic distribution of the sum of Bernoulli distributed random variables. To show the power of this technique, this section presents two competitive approaches to compute the probability of a database objects to be a hot item. While the first algorithm is a straightforward baseline algorithm, the second algorithm is an adaption of an algorithm proposed in [27].

4.6.1 Brute-Force Algorithm

That condition probability that an object $U \in \mathcal{DB}$ is a hot item, given that U is at position x can be rewritten as follows:

$$P(U \text{ is a } \textbf{hot item} | U = x) =$$

$$P(|\{U' \in \mathcal{DB} \setminus \{U\} | dist(x, U') < \epsilon\}| \geq minItems) =$$

$$\sum_{\substack{S_{minItems} \subseteq \mathcal{DB} \setminus \{U\} \\ |S_{minItems}| \geq minItems}} \left(\prod_{U' \in S_{minItems}} P(dist(x, U') \leq \epsilon) \cdot \prod_{U' \in \mathcal{DB} \setminus (S_{minItems} \cup \{U\})} (1 - P(dist(x, U') \leq \epsilon)) \right).$$

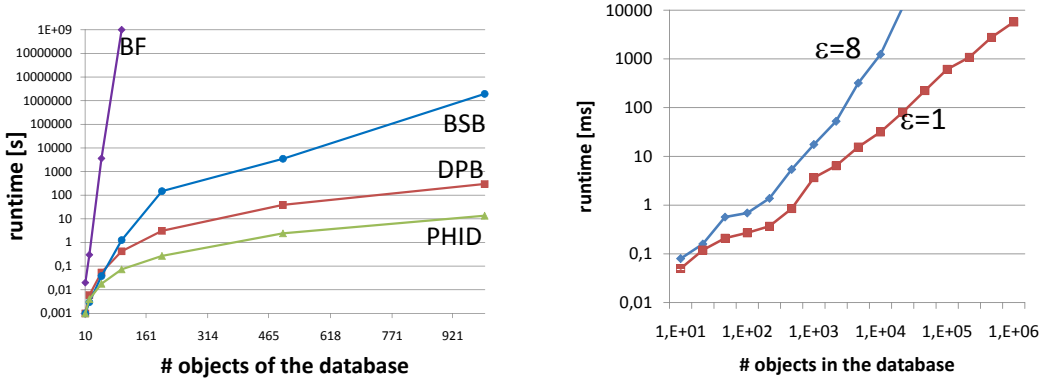
This computation is very expensive, since a total of $\sum_{i=minItems}^{|\mathcal{DB}|-1} \binom{|\mathcal{DB}|-1}{i}$ different sets $S_{minItems}$ have to be taken into account to calculate the sum term. Furthermore, for each summand we have to compute the product of $|\mathcal{DB}| - 1$ multipliers. Even if we ignore the cost of the product, the computational complexity of the sum term remains $O(2^{|\mathcal{DB}|})$. However, only those objects for which the probability that the predicate $dist(x, U') \leq \epsilon$ is satisfied is greater than zero have to be taken into account. In total, the computational complexity is $O(2^{|\mathcal{DB}'|})$, where $\mathcal{DB}' \subseteq \mathcal{DB}$ denotes the set of objects $U' \in \mathcal{DB}'$ for which $P(dist(x, U') \leq \epsilon) > 0$ holds. Even if $|\mathcal{DB}'| << |\mathcal{DB}|$, the computational cost would explode for reasonably large database size and reasonable settings for the $minItems$ value. Yet, this approach is faster than the naive approach that enumerates all possible worlds, whose run-time equals the case where $|\mathcal{DB}'| = |\mathcal{DB}|$.

4.6.2 Bisection-Based Algorithm

The computational cost can be significantly reduced if we utilize the bisection-based algorithm as proposed in [27]. The bisection-based algorithm uses a divide-and-conquer approach to compute the probability $P(|\{U' \in \mathcal{DB} \setminus \{U\} | dist(x, U') < \epsilon\}| = k)$ that exactly k objects are inside the query range by iteratively dividing the database $\mathcal{DB} \setminus \{I\}$ into two equally sized subsets $\mathcal{DB}_1$ and $\mathcal{DB}_2$, exploiting the law of total probability as follows:

$$P(|\{U' \in \mathcal{DB} \setminus \{U\} | dist(x, U') < \epsilon\}| = k) =$$

$$P(|\{U' \in \mathcal{DB}_1 | dist(x, U') < \epsilon\}| = i) \cdot P(|\{U' \in \mathcal{DB}_2 | dist(x, U') < \epsilon\}| = k - i)$$



(a) Evaluation of competing techniques.

(b) Scalability experiments.

Figure 4.4: Performance w.r.t database size.

This approach allows to reduce the effective size of the database relevant for each probability computation to $\frac{1}{2^a}$, but incurring a total of k^a probability computations rather than a single one, where a is the number of divide-and-conquer iterations. For each probability computation, this approach of [27] applies a straight-forward computation having exponential worst-case complexity in the size of the size of the database considered for the probability computation.

This technique can easily be adapted for our problem. The main idea is to recursively perform a binary split of the set of relevant objects, i.e. objects which have to be taken into account for the probability computation. Details of this algorithm can be found in [27].

4.6.3 Run-Time Experiments

In this section, we present the results of an experimental evaluation of the proposed methods w.r.t. efficiency.

Datasets and Experimental Setup

The hot item detection methods were applied to one artificial dataset (ART) and two scientific real-world datasets ($SCI1$,$SCI2$), based on the discrete uncertainty model.

In the ART dataset, each object is represented by a set of positions sampled uniformly from the $[0, 1]^5$ space.

Each of the 1500 objects of the datasets $SCI1$ and $SCI2$ consists of 10 samples, where each sample corresponds to a set of environmental sensor measurements of one single day that consist of several dimensions (attributes). The attribute set of $SCI1$ describes temperature, humidity and CO concentration, whereas $SCI2$ has a larger set of attributes (temperature, humidity, speed and direction of wind as well as concentrations of CO, SO_2, NO, NO_2 and O_3).

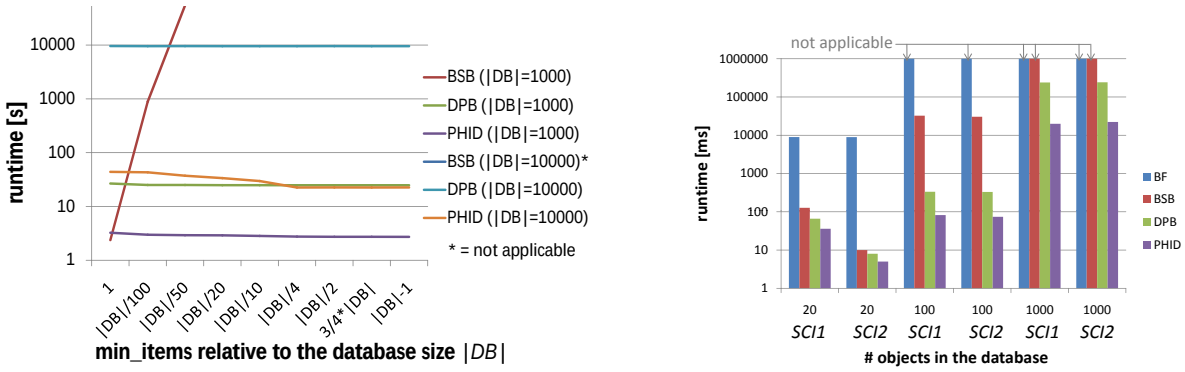
Here, we compare two variants of our approach denoted by *DPB* and *PHID*. The algorithm *DPB* applies the techniques presented in Section 4.5.1 on the complete database. In contrast, *PHID* applies a spatial pruning filter using the R*-Tree to find objects which must have a probability of zero to be in range of the query object. The performance of *PHID* and *DPB* is compared to that of the brute-force solution (*BF*) by applying the formula given in Definition 4.3. Furthermore, we compare them to the bisection-based method [27] (*BSB*) Let us note that *BSB* is considerably more efficient than the brute-force solution and, thus, a more challenging competitor than *BF*.

Note that in our algorithm, we concentrate on the evaluation of the CPU-cost only. The reason is that the $PHID$-algorithm is clearly CPU-bound. The only I/O bottleneck is the initial computation of the likelihood that U' is in the ϵ-range of u_i, for each object $U' \in DB$ and each sample u_i, where $U, U' \in DB$, $u_i \in U$ and $U \neq U'$. This requires a distance-range-self-join of the database which can be performed by a nested-block-loop join that requires $O(|DB|^2)$ page-faults in the worst case. In contrast, the CPU time for the $PHID$-algorithm is cubic: To compute the number of objects in ϵ-range of an instance of an object, i.e., to compute the sum of Bernoulli variables B_i, $1 \leq i \leq |\mathcal{DB}|$, we can either use the Poisson binomial recurrence technique presented in Section 3.4 or the generating function technique presented in Section 3.5. Either way, the incurred complexity is in $O(|DB|^2)$ time and has to be performed once for each sample in the database. In our experimental evaluation, we use an implementation of the Poisson binomial recurrence to compute the number of objects in the range of an instance of an object.

The first experiments relate to the scalability of the proposed approaches. The results depicted in Figure 4.4 demonstrate how the runtime of the competing techniques is influenced by the database size. Figure 4.4(a) shows that, though the bisection-based approach has exponential runtime, it outperforms the brute-force approach by several orders of magnitude. However, the Poisson binomial recurrence approaches scale significantly better than their competitors which in contrast to *DPB* and *PHID* have polynomial runtime. Furthermore, the pre-processing step of *PHID* obviously pays off. The performance can be further improved by an order of magnitude when applying the Poisson binomial recurrence only on objects U' having a non-zero chance to be in range. The next experiment shows the scalability of *PHID* for different ε-range values. Here, the average time required to compute the hot item probability for an object was measured. The results shown in Figure 4.4(b) demonstrate that *PHID* scales well, even for very large databases.

Figure 4.5(a) demonstrates the performance w.r.t. the $minItems$ value for different database sizes. Contrary to DPB and $PHID$, the BSB is very affected by the $minItems$ value due to the expensive probability computation. The slight increase of the DPB and $PHID$ performances can be explained by the reduced number of hot items with increasing $minItems$ value.

Finally, we evaluate the performance based on real-world data (cf. Figure 4.5(b)). Unlike the exponential algorithms, *DPB* and *PHID* are able to perform a full hot item scan of the database in reasonable time, even for a relatively large database size.



(a) Influence of *minItems*.

(b) Experiments on real-world data.

Figure 4.5: Performance experiments.

4.7 Conclusions

In this chapter, efficient solutions to answer probabilistic range queries on uncertain spatial data have been proposed. First, a solution for the simple case of a certain query point has been given. This solution, while solving a rather simple problem, was used as a showcase of how to apply the paradigm of equivalent worlds. Next, a solution for the more challenging case of uncertain query objects has been given. This solution accounts for the problem of stochastically dependent distances between uncertain objects. Finally, the problem of counting the number of uncertain objects in a region, and thus of the problem of finding hot items, i.e., objects for which at least *minItems* close objects exist, has been given an efficient solution. In particular this approach computes for each object o in an uncertain database the probability that o is a hot item. We proposed methods that are able to break down the high computational complexity required to compute for an object o the probability, that o is a hot item. We theoretically and experimentally show that our approach can efficiently find all hot items in $O(minItems \cdot n^2)$, where n is the number of uncertain objects and *minItems* is the number of objects required to be in proximity for an object to be considered a hot item. The decision if a single item is a hot item can be made in $O(minItems \cdot n)$.

In the next chapters, uncertain database solutions for more complex spatial similarity queries, including k-nearest neighbor queries (Chapter 6), similarity ranking queries (Chapter 7) and reverse k-nearest neighbor queries (Chapter 8) will be presented. All of these query types require, given two uncertain objects A and B, which object is closer to a another uncertain object R. The next section, will present a spatial pruning technique, to efficiently solve this problem for rectangular object approximations. This technique will then be utilized in the subsequent chapters.

Chapter 5

Optimal Spatial Pruning

Fast query processing of complex objects, e.g. spatial or uncertain objects, depends on efficient spatial pruning of the objects' approximations, which are typically minimum bounding rectangles (MBRs). This section proposes a novel effective and efficient criterion to determine the spatial topology between multi-dimensional rectangles. Given three rectangles R, A, and B in a multi-dimensional space, the task is to determine whether A is definitely closer to R than B. This *domination* relation is used in many applications to perform spatial pruning. Traditional techniques apply spatial pruning based on minimal and maximal distance. These techniques however show significant deficiencies in terms of effectivity. It is proven that the presented decision criterion is correct, complete, and efficient to compute even for high dimensional databases. Experiments show that the new pruning criterion, albeit very general and widely applicable, significantly outperforms current state-of-the-art pruning criteria.

5.1 Introduction

Speeding-up queries using minimal bounding rectangles (MBRs) as object approximations is a common technique used in many different ways. For example, rectangles are used for data sets with spatially extended objects such as polygons or CAD models because operations on the exact object representation are usually much more expensive than on the object approximations. Furthermore, MBRs are used as spatial key for spatial access methods, e.g. the most prominent ones including the R-Tree [82], R*-Tree [15], X-Tree [19] as well as specialized adaptations like the TPR tree [160] and the U-Tree [177] among many others. In the last decade, MBR approximations have also become very popular for uncertain databases [31, 45, 48, 51, 131] in order to approximate all possible locations of an uncertain vector object such as a GPS signal.

Rectangular approximations are commonly integrated into spatial pruning methods in order to speed-up spatial queries such as distance-range (ϵ-range) queries and k-nearest neighbor queries. Generally, current spatial pruning methods utilize the boundaries of regions, in particular of axis-aligned rectangles, in order to facilitate the pruning, i.e.

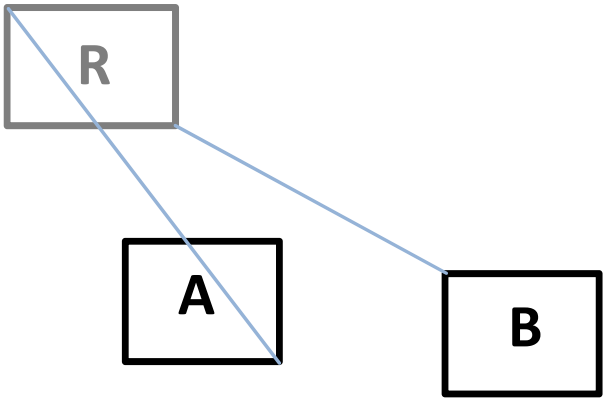


Figure 5.1: Spatial pruning on MBRs.

to filter out true drops that do not match the query predicate. In this context, spatial pruning techniques are used for numerous application fields including searching in multi-dimensional vector spaces [75, 88, 159], similarity search in time series databases [102], query processing on spatio-temporal data [84, 181] and probabilistic query processing on uncertain data [31, 50, 177].

Most types of spatial/similarity queries used for the above mentioned applications, including k-nearest neighbor (kNN) queries, reverse k-nearest neighbor (RkNN) queries, and ranking queries, commonly require the following information. Given three point sets A, B, and R in a multi-dimensional space $\mathbb{R}^d$, e.g. representing MBRs, the task is to determine whether object A is definitely closer to R than B w.r.t. a distance function defined on the objects in $\mathbb{R}^d$. If this is the case, we say A *dominates* B with respect to (w.r.t.) R. An example of such a situation is depicted in Figure 5.1. The concept of domination is a central problem for most types of similarity queries including the ones mentioned above in order to identify true hits and true drops (pruning). For example, in case of a 1NN query around R, we can prune B if it is dominated by A w.r.t. R and for an R1NN query around R, we can prune B if A dominates R w.r.t. B.

The domination problem is trivial for point objects. However, applied to rectangles, the domination problem is much more difficult to solve. The problem is that the distance between two objects approximated by rectangles is no longer a single value but is represented by an interval. If two such distance intervals overlap, we cannot definitely detect whether one distance is smaller than the other. Traditionally, the minimal distance and maximal distance between rectangles are used to decide which object is closer to another object. For example, in a nearest-neighbor query we can prune all objects whose minimal distance to the query object exceeds the maximal distance between at least one other object and the query object. In fact, the traditional distance approximations based on minimal and maximal distance are not always suitable to determine the distance relationship between objects. An example is depicted in Figure 5.1 showing three objects A, B and R each approximated by rectangles. In order to decide whether object A is closer to R than object B, we cannot apply the minimal/maximal distances because the minimal distance between

B and R is smaller than the maximal distance between A and R. Here, the problem is that when comparing the maximal distance between A and R with the minimal distance between B and R we take two different positions of the object R into account. For the maximal distance between A and R, we assume that the object R is located at the upper left corner of its rectangle approximation. For the minimum distance between B and R we assume that the object R is located at the lower right corner of its rectangle approximation. However, since an object approximated by a rectangle cannot be located at different positions at the same time, the two distances between A and R and between B and R depend on each other. To the best of our knowledge, none of the existing work, except approaches for reverse k-NN queries [180, 64], take this dependency into account. In our example, in fact, it can be detected that object A is closer to R than to object B when taking the above mentioned conditions into account. This chapter claims the following contributions.

- We discuss current state-of-the-art decision criteria for the domination problem among rectangles focussing on their correctness, completeness, and efficiency.
- We propose a novel decision criterion for the domination problem among rectangles that is correct, complete, and can be efficiently computed.
- We present extensive experiments to evaluate our new pruning criteria in comparison to state-of-the-art approaches.

The remainder of this chapter is organized as follows: First, Section 5.2 formally defines the concept of spatial domination. Next, Section 5.3 reviews existing approaches to detect spatial domination. Section 5.4 introduces a novel domination decision criterion. It is shown theoretically that this new domination decision criterion is optimal, i.e., it is able to return true in all, and only in cases where the domination relation truly holds. Furthermore, it is shown that the current state-of-the-art approach does not satisfy this optimality property. This decision criterion is exploited in Section 5.5 to find groups of rectangles that are able to collaboratively dominate another object. Section 5.6 shows how to apply the concept of spatial domination for spatial query problems. Section 5.7 presents experimental results and Section 5.8 finally concludes this section.

5.2 The Problem of Detecting Spatial Domination

Let $\mathcal{DB} \subseteq \mathbb{R}^d$ be a database of d-dimensional points and *dist* be a distance function on objects in $\mathbb{R}^d$. In this chapter we will focus on the L_p norms as the most commonly used family of distance functions in the area of similarity search. Intuitively, our problem is the following. Given the point sets $A, B, R \subseteq \mathcal{DB}$, we want to decide if A "is definitely closer to" R than B to R w.r.t. the distance function *dist*. If this is the case, we say A *dominates* B w.r.t. R, denoted by the predicate $(A \prec_R B)$. In fact, we will focus on points sets that represent rectangles, e.g. minimum bounding rectangles (MBRs) because rectangles are the most prevalent form of approximations for sets of points representing more complex objects like page regions of directory nodes in spatial index structures, polygons, time series, uncertain objects, etc. (see above).

Definition 23 (Domination)**.** *Let $A, B, R \subseteq \mathbb{R}^d$ be rectangles. The rectangle A dominates B w.r.t. R iff for all points $r \in R$ it holds that every point $a \in A$ is closer to r than any point $b \in B$, i.e.*

$$\begin{aligned}&(A \prec_R B) \Leftrightarrow \\ &\quad \forall r \in R, \forall a \in A, \forall b \in B : dist(a, r) < dist(b, r)\end{aligned} \tag{5.1}$$

To evaluate the predicate $(A \prec_R B)$, Equation 5.1 is not very helpful because a rectangle contains an infinite number of points in $\mathbb{R}^d$ and it is simply not computable to test all triples $a \in A$, $b \in B$ and $r \in R$. Rather, a domination decision criterion $DDC(A, B, R)$ for the single domination relation is required, which should satisfy the following three properties:

- **Correctness:** if $DDC(A, B, R)$ returns *true* then A dominates B w.r.t. R, i.e.

$$DDC(A, B, R) \Rightarrow (A \prec_R B).$$

- **Completeness:** if $DDC(A, B, R)$ returns *false* then A does not dominate B w.r.t. R, i.e.

$$\neg DDC(A, B, R) \Rightarrow \neg(A \prec_R B).$$

- **Efficiency:** $DDC(A, B, R)$ can be evaluated efficiently.

5.3 Existing Approaches

In the following, $X_i = [X_i^{min}, X_i^{max}]$ represents the interval of the rectangle X in dimension i, $X_i^{mid} = 1/2 \cdot (X_i^{min} + X_i^{max})$ is the mean of interval X_i, and x_i denotes the value of point x in dimension i $(1 \leq i \leq d)$.

5.3.1 The Min-/MaxDist decision criterion.

Probably the most well-known decision criterion for the domination problem among rectangles used in many database applications is based on two well known metrics defined on rectangles [159]. The minimum distance $MinDist(A, B)$ between two rectangles A and B always underestimates the distance of point pairs $(a, b) \in A \times B$ and is defined as

$$MinDist(A,B) := \sqrt[p]{\sum_{i=1}^{d} \begin{cases} |A_i^{min} - B_i^{max}|^p, \text{ if } A_i^{min} > B_i^{max} \\ |B_i^{min} - A_i^{max}|^p, \text{ if } B_i^{min} > A_i^{max} \\ 0 \qquad\qquad\quad , \text{ else} \end{cases}} = \sqrt[p]{\sum_{i=1}^{d} MinDist(A_i, B_i)} \tag{5.2}$$

The maximum distance $MaxDist(A, B)$ between two rectangles A and B always overestimates the distances of all point pairs $(a, b) \in A \times B$ and is defined as:

$$MaxDist(A,B) := \sqrt[p]{\sum_{i=1}^{d} \begin{cases} |A_i^{max} - B_i^{min}|^p, \text{ if} A_i^{mid} \geq B_i^{mid} \\ |B_i^{max} - A_i^{min}|^p, \text{ if} B_i^{mid} > A_i^{mid} \end{cases}} = \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, B_i)} \tag{5.3}$$

Definition 24 (Min-/MaxDist criterion)**.** *Let A, B, $R \in \mathbb{R}^d$ be rectangles. The* Min-/MaxDist domination decision criterion *is defined as*

$$DDC_{MinMax}(A, B, R) \Leftrightarrow MaxDist(A, R) < MinDist(B, R).$$

Lemma 7. *The Min-/MaxDist decision criterion is correct, i.e.,*

$$DDC_{MinMax}(A, B, R) \Rightarrow (A \prec_R B).$$

Proof. The following holds due to the conservative properties of *MinDist* and *MaxDist*: $DDC_{MinMax}(A, B, R) \Leftrightarrow MaxDist(A, R) < MinDist(B, R) \Rightarrow \forall a \in A, \forall r \in R, \forall b \in B : dist(a, r) \leq MaxDist(A, R) < MinDist(B, R) \leq dist(b, r) \Leftrightarrow (A \prec_R B)$. □

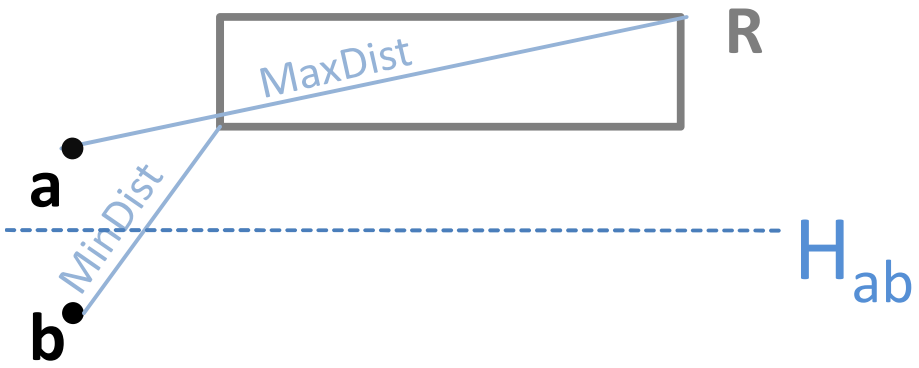


Figure 5.2: MBR pruning example

Lemma 8. *The Min-/MaxDist decision criterion is not complete, i.e.,*

$$\neg DDC_{MinMax}(A, B, R) \not\Rightarrow \neg(A \prec_R B).$$

Proof. Figure 5.2 shows an example for the 2D space where $DDC_{MinMax}(A, B, R)$ is false although $(A \prec_R B)$ holds. In the examples, $A = a$ and $B = b$ are rectangles with zero extension, i.e. points. Clearly, $MaxDist(a, R) < MinDist(b, R)$ is not satisfied, i.e. $DDC_{MinMax}(a, b, R)$ is false. The Voronoi line H_{ab} between a and b, i.e. the line containing all points that have equal distance to a and b, which is the dashed line in Figure 5.2 divides the 2D space into two half spaces. It is obvious that all points above that line (located in the half space containing a) have a distance to a that is smaller than the distance to b. Thus, according to Definition 23, a dominates b w.r.t. all objects which lie completely above H_{ab}. As a consequence, $(a \prec_R b)$ holds. □

Let us note that the Min-/MaxDist domination decision criterion is complete for two arbitrary rectangles A and B if R is a point, i.e. R has no extension in all dimensions. In addition, the Min-/MaxDist domination decision criterion can be computed efficiently in $O(d)$ time since the calculation of *MinDist* and *MaxDist* is linear in d.

5.3.2 Voronoi-based decision criterion.

The Voronoi plane H_{ab} between two points a and b that has been used in the proof of Lemma 8 is used in [180] as a different decision criterion for points. In a d-dimensional space $H_{ab} = \{x \in \mathbb{R}^d \mid dist(a, x) = dist(b, x)\}$ is a $(d-1)$-dimensional hyperplane containing all points having equal distance to a and to b. It divides the space into two half-spaces $H_{ab}(a)$ containing a and $H_{ab}(b)$ containing b. If a rectangle R lies completely within one of these half-spaces, then R is closer to the respective point in the same half-space. In the example of Figure 5.2, R is in the half-space $H_{ab}(a)$, thus all $r \in R$ are closer to a than to b. A Voronoi hyperplane between a point and a rectangle has been proposed in [64]. For the general case of two rectangles, we need to construct the Voronoi plane H_{AB}

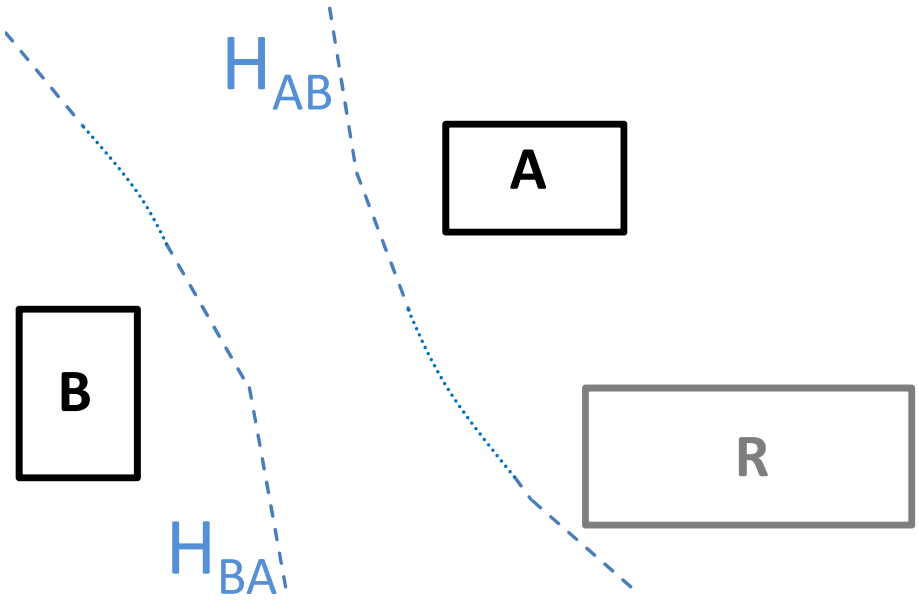


Figure 5.3: Voronoi-based decision criterion on MBRs

between two rectangles A and B which is the intersection of all Voronoi half-spaces between all pairs of points of the corresponding rectangles and can be defined as $H_{AB} = \{x \in \mathbb{R}^d \mid MinDist(x, B) = MaxDist(x, A)\}$, see [64]. An example of a Voronoi plane between two rectangles A and B is H_{AB}, depicted in Figure 5.3. This Voronoi plane is piecewise linear and curvilinear (again, see [64] for more details on the Voronoi plane between two rectangles). If a rectangle lies completely within the half-space $H_{AB}(A)$, then R is definitely closer to A. However, to determine the half-space containing all points that are definitely closer to B than to A, $H_{AB}(B)$ cannot be used and the Voronoi plane H_{BA} has to be computed. The reason is that unlike in the case of points, there exist points p for which neither $(A \prec_R p)$ nor $Dom(B \prec_R p)$ holds. Intuitively, the Voronoi-based domination decision criterion states that A dominates B w.r.t. R if R is completely contained in the half-space $H_{AB}(A)$.

Definition 25 (Voronoi-based criterion)**.** *Let A, B, $R \in \mathbb{R}^d$ be rectangles. The Voronoi-based decision criterion is defined as*

$$DDC_{\textbf{Voronoi}}(A, B, R) \Leftrightarrow R \subseteq H_{AB}(A).$$

Lemma 9. *The Voronoi-based decision criterion is correct and complete, i.e.,*

$$DDC_{\textbf{Voronoi}}(A, B, R) \Leftrightarrow (A \prec_R B).$$

Proof. By definition of H_{AB} the statement holds:
$DDC_{\textbf{Voronoi}}(A, B, R) \Leftrightarrow R \subseteq H_{AB}(A) \Leftrightarrow \forall a \in A, b \in B\colon\ R \subseteq H_{ab}(a). \Leftrightarrow \forall a \in A, \forall b \in B, \forall r \in R : dist(a, r) < dist(b, r) \Leftrightarrow (A \prec_R B).$ □

Table 5.1: Overview decision criteria

Criterion	Correct	Complete	Efficient
$DDC_{\mathbf{MinMax}}$	YES	NO	YES: $O(d)$
$DDC_{\mathbf{Voronoi}}$	YES	YES	NO: $O(2^d)$
$DDC_{\mathbf{Corner}}$	YES	YES	NO: $O(2^d)$
$DDC_{Optimal}$	YES	YES	YES: $O(d)$

Computing any Voronoi plane between any $a \in A$ and $b \in B$ to obtain the curvilinear plane as depicted in Figure 5.3 is rather complex. To the best of our knowledge, there exists no efficient solution for this problem. However, it is clear that any such algorithm must scale exponentially in the dimensions, since even for the simple case where b is a point, the number of different pieces of the plane is equal to the number of corners of A which is in $O(2^d)$ (cf. [64] for a discussion on the computation of such Voronoi planes).

5.3.3 Corner-based decision criterion.

The corner-based decision has recently been proposed as a pruning criterion for RkNN search of spatial objects in R^2 [64]. This approach exploits the property that the side $H_{AB}(A)$ of H_{AB} that is responsible for pruning is convex for RkNN queries. Thus, if a rectangle R is not fully contained in $H_{AB}(A)$ (i.e. R cannot be pruned), then at least one corner of R must be contained in $H_{AB}(B)$. Therefore, it is sufficient to consider only corners of MBRs. The Min-/MaxDist decision criterion, that is correct and complete in the case where only points are considered, is then applied to the corners. For more details on this decision criterion, refer to [64]. Since this criterion requires to consider all 2^d corners of MBRs, the complexity must scale in $O(2^d)$.

5.3.4 Summary.

Table 5.1 summarizes the discussion of existing decision criteria for the domination problem. It can be observed, that none of these approaches meets all the desired properties, i.e. either is not complete or suffers from exponential runtime. The fourth approach in Table 5.1 called "Optimal" is our new decision criterion which is described in the next section.

5.4 A Correct, Complete, and Linear-Time Domination Decision Criterion

We will derive a new decision criterion that is correct, complete, and can be computed in $O(d)$ time. Our novel domination decision criterion can be derived from the original definition of domination in Definition 23 by applying the following six equivalences.

Equivalence 1.

$$\begin{aligned} \forall a \in A, b \in B, r \in R &: dist(a,r) < dist(b,r) \Leftrightarrow \\ \forall r \in R &: MaxDist(A,r) < MinDist(B,r) \end{aligned}$$

Proof.
(1) "⇒"
If the left-hand side holds for each $r \in R$ then it also holds for that $a \in A$ and $b \in B$ that maximize and minimize the distance to r, respectively. These points $a \in A$ and $b \in B$ obviously determine the values of *MaxDist* and *MinDist*, respectively.

(2) "⇐"
If the right-hand side holds for each $r \in R$ as well as for that $a \in A$ and $b \in B$ that maximizes and minimizes the distance to r, i.e. determines the value of *MaxDist* and *MinDist*, respectively, then it also holds for any $a \in A$ and any $b \in B$. □

Equivalence 2.

$$\forall r \in R : MaxDist(A,r) < MinDist(B,r) \Leftrightarrow$$

$$\forall r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p}$$

Proof. Straightforward substitution of the definition of *MaxDist* and *MinDist* for L_p norms as given in Section 24. □

Equivalence 3.

$$\forall r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p} \Leftrightarrow$$

$$\forall r \in R : \sum_{i=1}^{d} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

Proof.

$$\forall r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p} \Leftrightarrow$$

$$\forall r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p < \sum_{i=1}^{d} MinDist(B_i, r_i)^p \Leftrightarrow$$

$$\forall r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - \sum_{i=1}^{d} MinDist(B_i, r_i)^p < 0 \Leftrightarrow$$

$$\forall r \in R : \sum_{i=1}^{d} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) \ < \ 0$$

□

Equivalence 4.

$$\forall r \in R : \sum_{i=1}^{d} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0 \Leftrightarrow$$

$$\max_{r \in R} (\sum_{i=1}^{d} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p)) < 0$$

Proof. Instead of considering all possible $r \in R$, it is sufficient to consider only that point $r' \in R$ which maximizes the left-hand side of the inequality. If the inequality holds for this point r', then it obviously holds for all possible $r \in R$ and vice versa. □

The next equivalence requires the following lemma:

Lemma 10. *Let $F : \mathbb{R}^d \to \mathbb{R}$ be a function that is summed by treating each dimension independently, i.e. there exists a function $f : \mathbb{R} \to \mathbb{R}$ such that*

$$F(o) = \sum_{i=1}^{d} f(o_i)$$

Also, let $A \subseteq \mathbb{R}^d$ be a rectangle and

$$\sigma := \text{argmax}_{a \in A}(F(a))$$

be the object in A that maximizes F. Then, the following holds:

$$\max_{a \in A} (\sum_{i=1}^{d} f(a_i)) = \sum_{i=1}^{d} \max_{a_i \in A_i} (f(a_i))$$

Proof.

$$\max_{a\in A}(\sum_{i=1}^{d} f(a_i)) \overset{\text{Def } F(a)}{=} \max_{a\in A}(F(a)) \overset{\text{Def } \sigma}{=} F(\sigma)$$

$$\overset{\text{Def } F(a)}{=} \sum_{i=1}^{d} f(\sigma_i) \overset{\text{Def } \sigma}{=} \sum_{i=1}^{d} \max_{a_i\in A_i}(f(a_i))$$

□

Equivalence 5.

$$\max_{r\in R}(\sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0 \Leftrightarrow$$

$$\sum_{i=1}^{d} \max_{r_i\in R_i}(MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

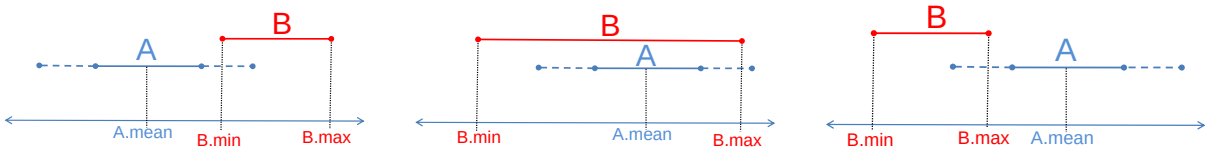
Proof. This follows from Lemma 10 by substituting

$$F(r) = MaxDist(A, r) - MinDist(B, r)$$

□

The final equivalence (equivalence 6) makes the equation computable. It is based on the assumption that for finding the maximum r_i in dimension i, it is sufficient to consider the boundary points (R_i^{min} and R_i^{max}) of the interval R_i. This assumption is proven in the following two lemmas.

Lemma 11. *Let A and B be intervals. The function $f : \mathbb{R} \to \mathbb{R}$ defined as $f(x) = MaxDist(A, x)^p - MinDist(B, x)^p$ has no local maximum.*



(a) $A.mean \leq B.min$ (b) $B.min < A.mean < B.max$ (c) $A.mean \geq B.min$

Figure 5.4: Illustration of Lemma 11.

Proof. By definition (c.f. Equations 5.2 and 5.3), the value of $MaxDist(A, x)$ only depends on $A.mean$ while the value of $MinDist(B, x)$ only depends on $B.min$ and $B.max$. Since $A.min \geq A.max$, this leads to the three possible cases depicted in Figure 5.4. In the

following, we will show for each of the cases that there may not exist any point for which it holds that $f(x)$ is increasing to the left of x and decreasing to the right of x.

Case 1: $A.mean \leq B.min$ (Figure 5.4(a)):

- For the interval $]-\infty, A.mean]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.max)^p - dist(x, B.min)^p =$
  $(|A.max - x|)^p - (|B.min - x|)^p$.
  This is constant if $p = 1$ or if $A.max = B.min$ since both $|A.max - x|$ and $|B.min - x|$ are decreasing. If $p > 1$, this term can be either **monotonically increasing or monotonically decreasing**, depending on whether $A.max$ is greater than $B.min$ or not.

- For the interval $]A.mean, B.min]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.min)^p - dist(x, B.min)^p$, where dist(x,A.min) is increasing and dist(x,B.min) is decreasing, thus $f(x)$ is **monotonically increasing** for any p.

- For the interval $]B.min, B.max]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.min)^p - 0$.
  This term is **monotonically increasing** since $dist(x, A.min)$ is increasing.

- For the interval $]B.max, \infty[$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.min)^p - dist(x, B.max)^p = (|A.min - x|)^p - (|B.max - x|)^p$.
  This is constant for $p = 1$ since both $|A.min - x|$ and $|B.max - x|$ are increasing. Since $A.min < A.mean < B.min < B.max$, above term is **monotonically increasing** for $p > 1$.

Putting the monotonic pieces together, it is clear that $f(x)$ may have one local minimum at $A.mean$, but definitely has no local maximum.

Case 2: $B.min < A.mean < B.max$ (Figure 5.4(b)):

- For the interval $]-\infty, B.min]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.max)^p - dist(x, B.min)^p =$
  $(|A.max - x|)^p - (|B.min - x|)^p$. This term is constant for $p = 1$ since both $|A.max - x|$ and $|B.min - x|$ are decreasing. Since $|A.max - x| \geq |A.mean - x| > |B.min - x|$, above term is **monotonically decreasing** for $p > 1$

- For the interval $]B.min, A.mean]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.max)^p - 0$. This is **monotonically decreasing** for any $p \geq 1$.

- For the interval $]A.mean, B.max]$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.min)^p - 0$. This is **monotonically increasing** for any $p \geq 1$.

- For the interval $]B.max, \infty[$ Equations 5.2 and 5.3 yield:
  $f(x) = dist(x, A.min)^p - dist(x, B.max)^p =$
  $(|A.min - x|)^p - (|B.max - x|)^p$.. This is constant for $p = 1$ since both $|A.min - x|$

and $|B.max - x|$ are increasing. Since $|A.min - x| \leq |A.mean - x| < |B.min - x|$, above term is **monotonically increasing** for $p > 1$.

Thus, $f(x)$ has one local minimum at $A.mean$, but definitely no local maximum.

Case 3: $A.mean \geq B.max$ (Figure 5.4(c)): This case is analogous to the first case. □

Lemma 12. *Let $f : \mathbb{R} \to \mathbb{R}$ be a function that has no local maximum and $I = [I_{min}, I_{max}] \subset \mathbb{R}$ be an arbitrary finite interval. The value that maximizes f in the interval I must be either I_{min} or I_{max}, i.e.*

$$\underset{i \in I}{argmax}(f(i)) \in \{I_{min}, I_{max}\}$$

Proof. Let $p \in [I_{start}, I_{end}]$ be the value that maximizes f in I, i.e. $p = \text{argmax}_{i \in I}(f(a))$. Then, $\forall i \in I : f(i) \leq f(p)$, in particular, $f(I_{min}) \leq f(p)$ and $f(I_{max}) \leq f(p)$. Note that $f(I_{min}) < p$ and $f(I_{max}) < p$ cannot both be true, because this would be a contradiction to the assumption that $f(x)$ has no local maximum. Thus it must either hold that $f(I_{start}) = f(p)$ or $f(I_{end}) = f(p)$, i.e. $I_{min} = \text{argmax}_{i \in I}(f(x))$ or $I_{max} = \text{argmax}_{i \in I}(f(x))$. □

Now we can derive the final equivalence.

Equivalence 6.

$$\sum_{i=1}^{d} \max_{r_i \in R_i}(MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0 \Leftrightarrow$$

$$\sum_{i=1}^{d} \max_{r_i \in \{R_i^{min}, R_i^{max}\}}(MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

Proof. Follows from Lemma 11 and Lemma 12. □

Definition 26 (optimal decision criterion)**.** *Our novel optimal domination decision criterion is defined as*

$DDC_{Optimal}(A, B, R) \Leftrightarrow$

$$\sum_{i=1}^{d} \max_{r_i \in \{R_i^{min}, R_i^{max}\}}(MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

Lemma 13. *The novel optimal domination decision criterion is correct and complete.*

Proof. Correctness and completeness follow directly from equivalences 1 to 6. □

Obviously, the novel optimal domination decision criterion can be computed in $O(d)$ time and, thus, satisfies all three desired properties described in Section 5.2.

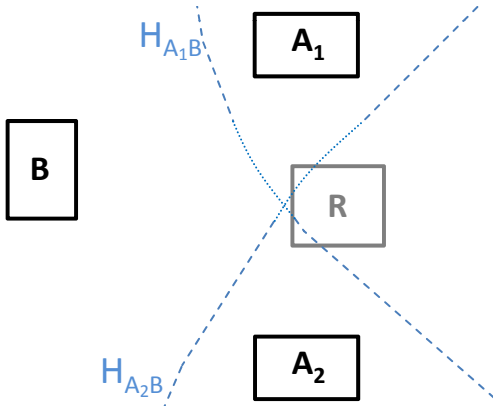


Figure 5.5: Partial Domination example for an RNN-query

5.5 Domination Count Computing

In most applications, testing the single domination relation of only two rectangles (w.r.t. a reference rectangle) is too basic. Rather, in the context of a set of rectangles $\mathcal{O} \subseteq \mathbb{R}^d$, the number of rectangles $A_i \in \mathcal{O}$ that dominate a given rectangle B w.r.t. R (referred to as *domination count*) is required. For example, a kNN query algorithm can use the information that at least k rectangles of $\mathcal{O}$ dominate rectangle $B \in \mathcal{O}$ w.r.t. a query rectangle R to identify B as true drop that can be pruned. The number of rectangles that dominate a given rectangle can analogously be used e.g. for RkNN queries and inverse ranking queries.

Definition 27 (domination count)**.** *Let $B, R \subseteq \mathbb{R}^d$ be rectangles and $\mathcal{O}$ be a set of rectangles. The* domination count *of B w.r.t. R is defined by:*

$$DC(\mathcal{O}, B, R) = \min_{r \in R}\{|A_i \in \mathcal{O} : MaxDist(A_i, r) < MinDist(B, r)|\}$$

Intuitively, if the domination count of B w.r.t. R is k, then for each point $r \in R$ there exist at least k rectangles $A_i \in \mathcal{O}$ which are closer to r than B.

Let us note that the domination count of B w.r.t. R cannot be computed by simply counting the number of rectangles that dominate B w.r.t. R by means of Definition 23 because this does not involve groups of rectangles that dominate R collectively, but not individually. An example of such a group of rectangles is shown in Figure 5.5. Neither rectangle A_1 nor rectangle A_2 dominates B w.r.t. R. However, B is dominated *partially* by A_1 and partially by A_2, respectively, i.e. it is dominated by A_1 and A_2 w.r.t. specific subregions of R.

However, when considering any point $r \in R$, rectangle B is dominated by at least one of the two rectangles A_1, A_2 w.r.t. r and, thus, B is dominated by the group $\mathcal{A} = \{A_1, A_2\}$ according to Definition 23.

In general, the problem of finding the subregion with the minimal domination count is hard. First, the computation of the intersection of a half-space and a hyper-polyhedron

becomes increasingly complex [180] for increasing dimensionality. Secondly, the number of subregions grows very fast. To give a brief intuition of the possible number of subregions generated by a total of n objects, consider the case of axis parallel pruning regions. If $n \leq d$, then each object may split R in a different dimension, resulting in a total of 2^n subregions. For $n > d$, balanced splitting of dimensions results in at least $(1 + \lfloor \frac{n}{d} \rfloor^d)$ subregions. If d is assumed to be constant, then $(1 + \lfloor \frac{n}{d} \rfloor^d) \in O(n^d)$. Thirdly, the resulting subregions can be complex d-dimensional polygons, particularly the subregions could have not only straight sides but also parabolic sides which makes computations involving these polygons very complex.

Though we are not able to compute the exact domination count of a given rectangle efficiently, we can try to find efficient solutions for approximating the domination count of a rectangle. In principal, in order to determine the domination count of B w.r.t. R we need to take the two constituting types of dominations into account: The first part is to count all objects A for which $(A \prec_R B)$ holds. This number is called *basic domination count*. This can be done using e.g. $DDC_{Optimal}$. The second and more challenging part is to detect all minimal groups $\mathcal{A}$ that dominate B as a group but do not contain an element that already dominates B separately, i.e. each $A_i \in \mathcal{A}$ only partially dominate B. The consideration of this type of domination requires the concept of *partial domination* which will be introduced later on.

A simple lower bound of the domination count can be achieved by computing the basic domination count. Intuitively, the basic domination count simply counts the number of rectangles that (completely) dominate the rectangle B w.r.t. rectangle R, i.e. neglects groups of rectangles that only partially dominate B separately but completely dominate B as a group.

Definition 28 (Basic Domination Count)**.** *Let $\mathcal{O} = \{A_1, ..., A_N\}$ be a set of d-dimensional rectangles and let $B, R \subseteq \mathbb{R}^d$ be two rectangles. The* basic domination count *of B w.r.t. R is the number of objects in $\mathcal{O}$ that dominate B w.r.t. R, formally:*

$$DC_{basic}(\mathcal{O}, B, R) = |\{A_i \in \mathcal{O} \,|\, (A_i \prec_R B)\}|.$$

Using our novel domination decision criterion $DDC_{Optimal}$, the basic domination count DC_{basic} can be computed in $O(N \cdot d)$. This is worth noting since existing decision criteria only allow either to compute the exact DC_{basic} value in exponential time or to compute an approximation of the DC_{basic} value in linear time. In the latter case, we would obtain a lower bound of DC_{basic} which makes the lower bounding estimation of the domination count even more loose.

As discussed above, the domination count also takes into account all sets of rectangles that increase the domination count of a rectangle as a group and that do not contain any element that does so separately. Therefore, we need the concept of *partial domination*. In the remainder of this section, we will first formalize the concept of partial domination. In particular we will discuss how our novel domination decision criterion $DDC_{Optimal}$ can be used for (i) detecting partial domination and (ii) deriving a conservative approximation of the domination count.

5.5.1 Partial Domination

The concept of *partial domination* (cf. Figure 5.5) was first introduced in [64] (in this work, we used the term "partial pruning") for boosting $RkNN$ queries in the 2D space. It can be applied to any other similarity query type analogously.

Definition 29 (partial domination)**.** *Let $A, B, R \subseteq \mathbb{R}^d$ be rectangles.* A dominates B partially *w.r.t. R, denoted by $PDom(A, B, R)$ if A dominates B for some, but not all $r \in R$, i.e.*

$PDom(A, B, R) \Leftrightarrow$

$$\neg(\forall a \in A, b \in B, r \in R : dist(b, r) > dist(a, r)) \tag{5.4}$$

$$\bigwedge$$

$$\exists r \in R : \forall a \in A, b \in B : dist(b, r) > dist(a, r) \tag{5.5}$$

Inequality 5.4 holds if A does not dominate B w.r.t. all points $r \in R$. Note that Inequality 5.4 is simply the negation of $(A \prec_R B)$ and can also be computed in $O(d)$ using our novel decision criterion $DCC_{Optimal}$. Inequality 5.5 is only satisfied if there exists an $r \in R$ for which B is dominated by A.

Obviously, the sets of objects that dominate B as a group can only contain rectangles A_i that partially dominate B, i.e. for which $PDom(A_i, B, R)$ holds. In other words, for the computation of the second part of the domination count of a rectangle B, we could use the detection of partial domination as a first step because only those rectangles A_i for which $PDom(A_i, B, R)$ holds could be the elements of those set of rectangles that dominate B as a group.

Partial domination can efficiently be detected by applying the following six equivalences analogously to Section 5.4. We start with above Inequality 5.5.

Equivalence 7.

$$\begin{aligned} \exists r \in R : \forall a \in A, b \in B &: dist(b, r) > dist(a, r) \\ \Leftrightarrow \exists r \in R &: MaxDist(A, r) < MinDist(B, r) \end{aligned}$$

Proof. This proof is analogous to the proof of Equivalence 1, i.e. it exploits that the $DDC_{\mathbf{MinMax}}$, decision criterion is optimal in the case where R is a point. □

Equivalence 8.

$$\exists r \in R : MaxDist(A, r) < MinDist(B, r) \Leftrightarrow$$

$$\exists r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p}$$

Proof. Follows directly from the definition of *MaxDist* and *MinDist* for L_p norms. □

Equivalence 9.

$$\exists r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p} \Leftrightarrow$$

$$\exists r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p < 0$$

Proof.

$$\exists r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p} \Leftrightarrow$$

$$\exists r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p < \sum_{i=1}^{d} MinDist(B_i, r_i)^p \Leftrightarrow$$

$$\exists r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - \sum_{i=1}^{d} MinDist(B_i, r_i)^p < 0 \Leftrightarrow$$

$$\exists r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p < 0$$

□

Equivalence 10.

$$\exists r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p < 0 \Leftrightarrow$$

$$MIN_{r \in R}(\sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

Proof. The rationale for equivalence 10 is that if there exists an $r \in R$ for which the left-hand side returns less than 0, then this also holds for the r which minimizes the term on the right-hand side and vice versa. □

Equivalence 11.

$$\min_{r \in R}(\sum_{i=1}^{d} MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0 \Leftrightarrow$$

$$\sum_{i=1}^{d} \min_{r_i \in R_i} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

Proof. This proof is analogous to the proof of Equation 5 using minimization instead of maximization. □

Analogously to Equivalence 6, the last equivalence below makes the equation computable. Again, we need two lemmas.

Lemma 14. *Let D be a one dimensional vector database using L_p-Norm. Let A and B be intervals. The function $f : \mathbb{R} \to \mathbb{R}$:*

$$f(x) = maxDist(A, x)^p - minDist(B, x)^p$$

may have a local minimum only at A.mean.

Proof. This proof is contained in the formal proof of lemma 11. □

Lemma 15. *Let $f : \mathbb{R} \to \mathbb{R}$ be a function that has at most one local minimum at x. For any finite interval $I \subset \mathbb{R} = [I_{start}, I_{end}]$ the following holds:*

$$\underset{i \in I}{argmin}(f(i)) \in \{I_{start}, I_{end}, x\}$$

That is, the point of the interval I that minimizes $f(x)$ must be either the lower or the upper bound of I, or the local minimum x.

Proof. The proof is similar to the proof of Lemma 12 and thus omitted here. □

In consideration of the above lemmas we now derive the final equivalence:

Equivalence 12.

$$\sum_{i=1}^{d} \min_{r_i \in R_i} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0 \Leftrightarrow$$

$$\sum_{i=1}^{d} min_{r_i}(MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0, \qquad where\ r_i \in \{R_i^{min}, R_i^{max}, A_i^{mid}\}$$

Proof. Directly follows from Lemma 14 and Lemma 15. □

Thus, using the formula in Equivalence 12 we can efficiently detect all partial dominations. However, as indicated above, this is only the first step towards computing the domination count. In fact, we need to determine that subregion of the reference rectangle R, for which the domination count is minimal. Since we cannot test all possible points $r \in R$ (see also the discussion above), we propose three heuristics to conservatively approximate the domination count of a rectangle.

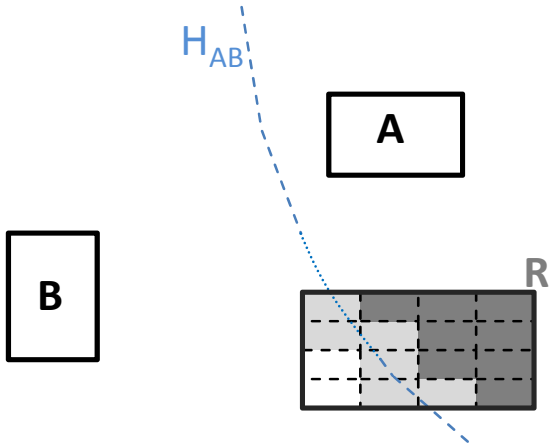


Figure 5.6: Partial domination using grid partitioning

5.5.2 Domination Count Estimation

Using the techniques proposed in Sections 5.4 and 5.5.1 we can check if an MBR A dominates B completely or partially w.r.t. R. These tests are generally applicable as long as the involved objects are MBRs. For calculating the domination count of B it is therefore possible to split R into smaller MBRs and then calculate the domination count for each cell individually. The following three heuristics use different approaches for splitting R to estimate the domination count.

Domination Count Estimation based on grid partitioning

A straight forward approach for splitting R is performed by using a grid with a fixed number m of partitions in each dimension. Considering the example in Figure 5.6, we can (using the decision criteria for domination and partial domination) assert that A dominates B w.r.t. all dark gray cells and partially dominates B w.r.t. all light gray cells of R. For the rest of the cells (white) A does not dominate B. Using this grid partitioning, the domination count ($DC(\mathcal{O}, B, R)$) can be estimated by the minimum domination count of all cells $c_i \in R$, that is:

$$DC_{grid}(\mathcal{O}, B, R) = min_i(DC_{basic}(\mathcal{O}, B, c_i))$$

This estimation is valid as we know that B is dominated by at least this amount of $A_i \subset \mathcal{O}$ w.r.t. each cell $c_i \in R$.

An example for the grid based partial pruning is given in Figure 5.7. Here an MBR R is partitioned into 16 cells. In addition two Voronoi hyperplanes H_{A_1B} and H_{A_2B} are shown. The objects $\mathcal{O} = \{A_1, A_2\}$ and B generating the hyperplanes are ommited here. For the area on the right-hand side of H_{A_1B}, object B is dominated by object A_1 and for the left-hand side of H_{A_2B}, B is dominated by A_2. It is clear that neither A_1 nor A_2 (fully-) dominate B with respect to the whole MBR R. For each cell the conservative domination

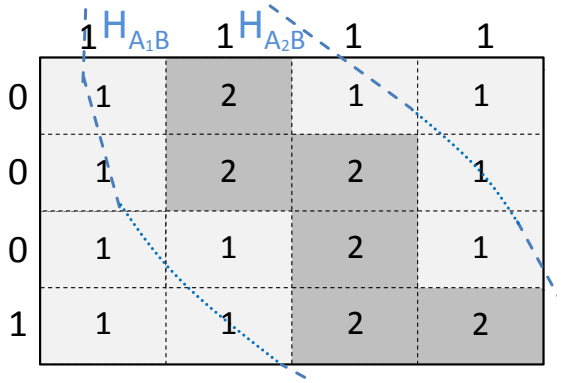


Figure 5.7: Domination Count estimation using grid partitioning.

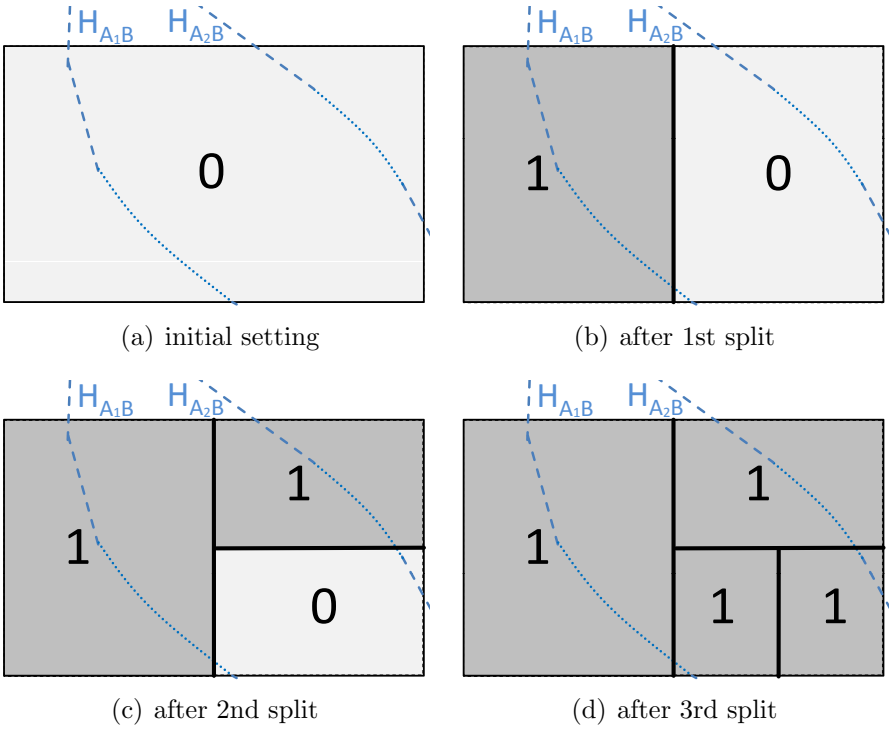
count $DC_{basic}(\mathcal{O}, B, c_i)$ is shown. With respect to dark cells, A_1 and A_2 dominate B and thus the cells have a value of 2. With respect to light cells, only one of the two objects dominates B, therefore they get marked with a value of 1. By taking the minimum value of all cells $c_i \in R$ we obtain $DC_{grid}(\mathcal{O}, B, R) = 1$. The advantage of this approach is, that it returns a very accurate estimation of the domination count while avoiding expensive materialization of the Voronoi hyperplanes. The accuracy can be boosted by increasing the number of splits per dimension. In return increasing m will highly increase the runtime of the algorithm, as the number of cells $c_i \in R$ is m^d. This implies that this approach is not applicable for high dimensions. For each cell c_i, $DC_{basic}(\mathcal{O}, B, c_i)$ can be computed in a single scan of the objects for which $PDom(A_i, B, R)$ holds using the $DDC_{Optimal}$ (c.f. Definition 26). Thus the total time complexity is in $O(d \cdot |\mathcal{O}| \cdot m^d)$.

Domination Count Estimation based on slices

In order to reduce the runtime of the domination count estimation, we propose a second algorithm, which is not based on a grid partitioning. Instead of cells, this approach considers *slices*. Therefore an MBR R is split into m *slices* s_i^{dim} in each of the d-dimensions ($1 \leq dim \leq d$). This results in $d \cdot m$ overlapping *slices*. The domination count $DC(\mathcal{O}, B, R)$ can then be approximated by computing, for each dimension, the minimal domination count of all slices and using the result of the dimension maximizing this estimation.

$$DC_{slice}(\mathcal{O}, B, R) = max_{dim}(min_i(DC_{Basic}(\mathcal{O}, B, s_i^{dim})))$$

For example, the domination count $DC(\mathcal{O}, B, s_i)$ for each slice s_i (i.e. each row and each column) and each cell c_i is shown in Figure 5.7 for a 2 dimensional MBR R. The minimal domination count considering all rows is 0, while the minimal domination count w.r.t. all columns is 1. Thus $DC_{slice}(\mathcal{O}, B, s_i) = 1$ in this example. The complexity of this algorithm is in $O(m \cdot d)$. However, this approach yields much worse results than the grid-based approach for an identical m parameter. Details can be found in our experiments (Section 5.7).



(a) initial setting (b) after 1st split

(c) after 2nd split (d) after 3rd split

Figure 5.8: Example for computing DC_{bisect}

Domination Count Estimation based on bisections

We next propose a bisection based approach that yields much better efficacy, while still being linear in d. This approach works iteratively. During each iteration, one section of R is chosen to be split evenly (mean split) in one dimension. After m splits, this results in $m+1$ sections $s_0 \cup s_1 \cup \ldots \cup s_m = R$ and it holds that:

$$DC_{bisect}(\mathcal{O}, B, R) = min_i(DC_{basic}(\mathcal{O}, B, s_i))$$

The challenge here is to wisely choose the split section of R and the dimension to split in each iteration.

We propose to split the section $s \subset R$ with the lowest domination count estimation. This decision is optimal, because the estimation of $DC(\mathcal{O}, B, R)$ is determined by the section which results in the lowest domination count. Thus, in order to increase the domination count approximation, s must be split. If the decision for s is ambiguous, then one of the candidates of s is chosen arbitrarily. To determine the split axis, the heuristic tests each dimension, and greedily uses the dimension that yields the highest domination count $DC_{bisect}(\mathcal{O}, B, R)$ considering the two resulting bisections of s. In the case of ties the axis is chosen which maximizes the sum $\sum_{i=0}^{m} DC_{basic}(\mathcal{O}, B, s_i)$. An example is shown in Figure 5.8. Considering Figure 5.8(a) it is clear that none of the two objects $A_1, A_2 \in \mathcal{O}$ that are responsible for the Voronoi hyperplanes H_{A_1B} and H_{A_2B} dominates B w.r.t. R. Beginning with the y-axis as split axis would result in two equi-sized MBRs both of which

result in a domination count $DC_{bisect}(\mathcal{O}, B, R)$ of 0 and therefore the approximation of $DC(\mathcal{O}, B, R)$ does not increase. Choosing the x-axis as split axis would result in two equi-sized MBRs shown in Figure 5.8(b) yielding the same domination count approximation but a higher sum ($\sum_{i=0}^{m} DC_{basic}(\mathcal{O}, B, s_i) = 1$). In the next iteration, the right MBR is chosen to be split, since it is responsible for the lowest domination count approximation. Both possible split axes are equal according to our heuristic. In the example, the y-axis is chosen arbitrarily (c.f. Figure 5.8(c)). The third (see Figure 5.8(d)) split of the lower-right MBR increases $DC_{bisect}(\mathcal{O}, B, R)$ to 1.

The bisection-based Domination Count Estimation algorithm uses m iterations. In each iteration i there exist exactly i sections of which the section with the lowest conservative domination count has to be found. This yields a complexity of $O(m^2)$ but can be reduced to $O(m \cdot log(m)))$ by using a Priority Queue to find the section with the lowest conservative domination count. For the greedy heuristic, in each iteration, each dimension has to be tested to determine the best split axis in $O(m \cdot d)$. Thus we get a total complexity of $O(m \cdot log(m) + m \cdot d) = O(m \cdot \max(log(m), d))$, where m is the number of iterations.

5.6 Boosting Similarity Queries

In this section, we will show how the concepts of domination and domination count can be used to boost the pruning power of similarity search algorithms.

Nearest-Neighbor Search. For a kNN query with query object Q, any object $O \in \mathcal{DB}$ can be pruned if $DC(\mathcal{DB}, O, Q) \geq k$. Note, that for a kNN query, the query object corresponds to the reference object R in Definition 26. Thus, $DDC_{Optimal}$ has an advantage over $DDC_{\mathbf{MinMax}}$ in the general case but is equivalent in the special case where Q is a point, because then $DDC_{\mathbf{MinMax}}$ is optimal. However, as discussed above, there are many applications in which the query object is a rectangle.

Reverse Nearest Neighbor Search. For a general RkNN query with query object Q, any object $O \in DB$ can be certainly pruned if $DC(\mathcal{DB}, Q, O) \geq k$. For R$k$NN queries, the query object corresponds to the object B in Definition 26. Thus $DDC_{Optimal}$ is superior to $DDC_{\mathbf{MinMax}}$ also in the special case where the query object is given as a point.

True hit detection. Our decision criterion $DDC_{Optimal}$ can be used to prune potential result candidates by being able to decide that they must not be part of the result set. A problem very similar to pruning is the detection of true hits, i.e. to quickly decide that a potential result candidate must be part of the result set. For example, in the case of kNN queries, an object B is a true hit, if there may be at most k objects that can be closer to R than B. In other words, B is a true hit, if it dominates at least $|\mathcal{DB}| - k$ objects. Thus, for a kNN query, an object B is a true hit if $|\{A \in \mathcal{DB} | (B \prec_Q A)\}| > |\mathcal{DB}| - k$. For a R$k$NN query, an object B is a true hit if $|\{A \in \mathcal{DB} | (Q \prec_B A)\}| \geq |\mathcal{DB}| - k$. The concept of partial domination can be applied to true hit detection as well.

Inverse Similarity Ranking. The problem of inverse ranking is to determine for a given query object Q the number of objects that are closer to a given reference object R. Such queries are useful e.g. to determine the financial standing of bank customers in relation to

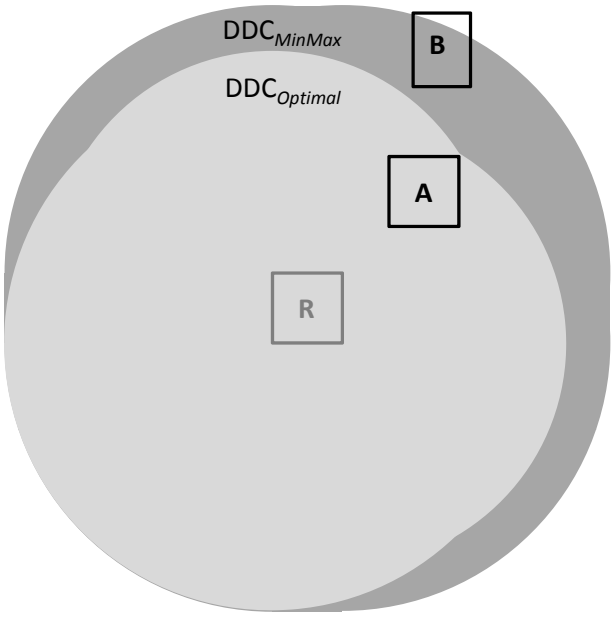


Figure 5.9: Refinement areas for fixed R and A

existing customers. In this scenario, the attributes of customers are often uncertain (e.g. income of $40k-50$k) and thus modeled by uncertain regions, i.e. rectangles. Lower and upper bounds for the rank of Q are $DC(\mathcal{DB}, Q, R)+1$ and $|\mathcal{DB}|-|\{A \in \mathcal{DB}|(Q \prec_R A)\}|+1$, respectively.

5.7 Experimental Evaluation

This section evaluates the effectiveness and efficiency of our novel domination decision criterion in comparison to the prevalent $DDC_{\mathbf{MinMax}}$ decision criterion. After that we evaluate the performance of our domination-count-detection approach which is based on the concept of partial domination. Finally, we exemplarily will show how our new methods influences the performance of existing similarity search methods designed for kNN and RkNN queries. For all experiments the underlying distance function is the euclidian norm.

5.7.1 Single Object Domination

We first evaluate the effectiveness gain of $DDC_{Optimal}$ compared to $DDC_{\mathbf{MinMax}}$ in consideration of the decision power. In order to measure the decision power, we take for a given pair of rectangles R and A the region into account containing all points that cannot be detected to be dominated by A w.r.t. R. In the reminder we call this region refinement area, since all objects intersecting this area might be refined in order to detect the domination relation. It should be clear that the smaller this area, the higher the corresponding domination power. Figure 5.9 exemplarily shows the refinement areas

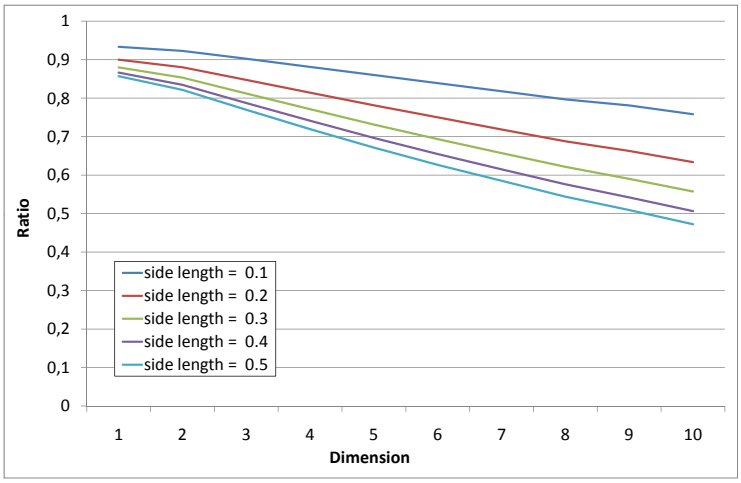
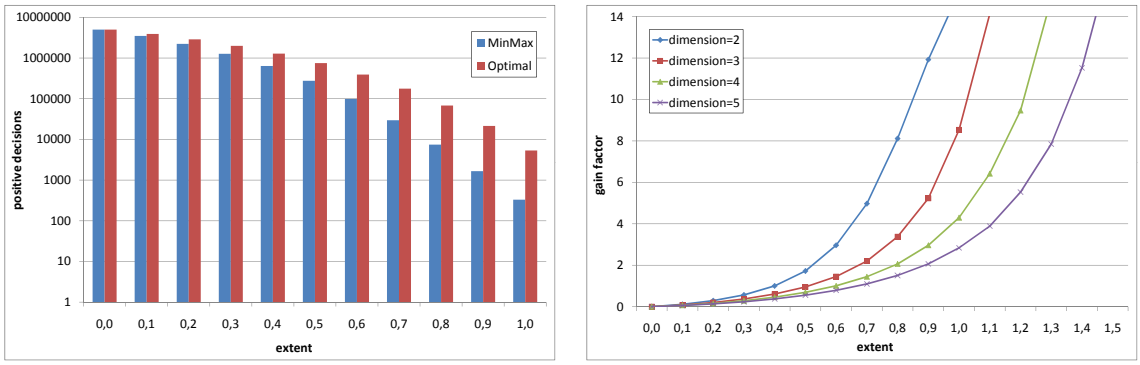


Figure 5.10: Ratio of the refinement areas of $DDC_{Optimal}$ and $DDC_{\mathbf{MinMax}}$ w.r.t. dimension and size of MBRs

for the 2-dimensional MBRs A and R w.r.t. both criteria $DDC_{\mathbf{MinMax}}$ and $DDC_{Optimal}$, respectively. In this example, object B is detected to be dominated by A only if we apply $DDC_{Optimal}$ instead of $DDC_{\mathbf{MinMax}}$. The refinement areas depend on several conditions such as position, shape, distance and extension of the MBRs specifying the refinement area as well as the dimensionality of the space. For our experiment evaluating the domination power, pairs of MBRs R and A are positioned in $[0,1]^d$ with a fixed $MinDist$ of 0.5 and equal distances in each dimension. The length of each side of the two MBRs was scaled from 0.1 to 0.5 and dimension screened from 1 to 10. The gain of the domination power is measured by the ratio of the volumes of the refinement area w.r.t. $DDC_{Optimal}$ and the refinement area w.r.t. $DDC_{\mathbf{MinMax}}$ by means of Monte-Carlo-Sampling. The results in Figure 5.10 show that $DDC_{Optimal}$ leads to a much higher decision power. The effect becomes more evident as the number of dimensions and the extension of the MBRs increase. As expected, increasing the extension of the MBRs leads to diminishing completeness of the $DDC_{\mathbf{MinMax}}$ decision criterion. It is notable, that the $DDC_{\mathbf{MinMax}}$ criterion suffers considerably from an increasing dimensionality. Note that we used MBRs of equal side length as we observed that this setting favors the decisions power based on $DDC_{\mathbf{MinMax}}$ in order to make a fair comparison. In fact, the advantage of the gain of the decision power based on $DDC_{Optimal}$ will increase even further for non-quadratic rectangles.

In addition to the above experiment which is more from a theoretical point of view, we compared the number of domination relations detected by applying $DDC_{Optimal}$ and $DDC_{\mathbf{MinMax}}$. Therefore we randomly generated one million triples of rectangles (A, B, R) with a fixed extent (i.e. the sum of side lengths) in the $[0,1]^2$ space. For each triple we tested if the decision criterion is able to determine whether $(A \prec_R B)$ holds. Finally we aggregated the number of positive decisions for different extents of the MBRs. The results are illustrated in Figure 5.11(a). Note that an extent of zero yields points instead of rectangles such that both criteria perform equal. However, we can observe that with increasing extent,



(a) Positive decisions made

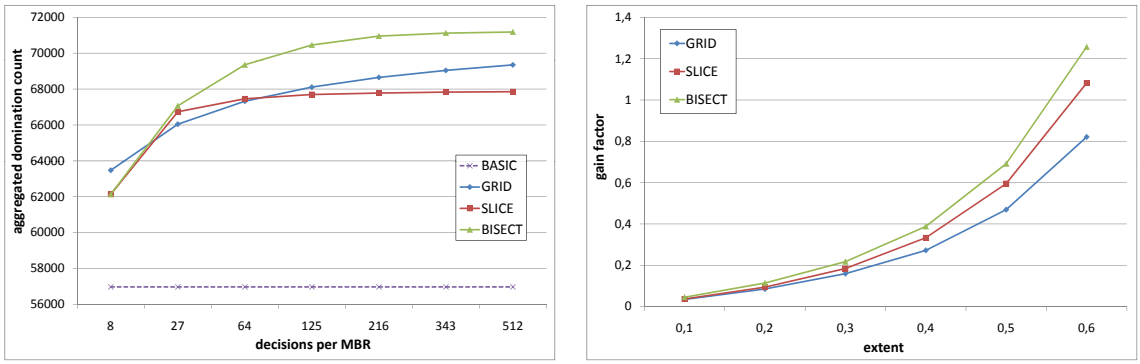
(b) Factor of positive decisions made more by the optimal criterion

Figure 5.11: Comparison of MinMax- and optimal-criterion on synthetic data

the percentage of positive decisions of $DDC_{Optimal}$ compared to $DDC_{\mathbf{MinMax}}$ increases considerably. The gain of the decision power based on $DDC_{Optimal}$ over $DDC_{\mathbf{MinMax}}$ is illustrated in Figure 5.11(b) showing the factor of positive domination decisions using $DDC_{Optimal}$ in comparison of that using $DDC_{\mathbf{MinMax}}$. We varied the dimensionality of the rectangle space up to 5 dimensions. Here we can observe that the gain increases with increasing extent. In contrast, when increasing the dimensionality, the gain of the decision power decreases. The reason is that in this setting, the extent of the MBRs is fixed for all dimensionality settings such that the average side length per dimension decreases and MBRs converge to points for high dimensionality.

5.7.2 Domination Count Estimation

The next experiments evaluate the accuracy of the domination count estimation of a rectangle B w.r.t. a rectangle R for the approaches proposed in Section 5.5.2: Basic Domination Count Estimation (DC_{basic}), grid partitioning (DC_{grid}), slice partitioning (DC_{slice}) and bisection based partitioning (DC_{bisect}). For these experiments, we generated one thousand three-dimensional MBRs with random positions. One MBR R was positioned in the center of the data space. Then we computed the conservative domination count w.r.t. R for each MBR using the four approaches mentioned above. We performed several runs for different parametric settings and averaged the results. Figure 5.12 shows the performance of all four approaches in terms of estimated domination count. First, we want to get a grasp of the relationship between accuracy of the domination count estimation and the cost required for the domination count computation. Therefore, the cost is measured in terms of number of calls of $DDC_{Optimal}$. It should be clear that when increasing the number of MBR partitions, and thus the required number of $DDC_{Optimal}$ calls, the estimation accuracy of all approaches improves, except for the basic approach since it does not use any partitioning. Figure 5.12(a) shows the results for MBRs with an extent of 0.3. It can be seen that all approaches show a significant improvement compared to DC_{basic} when increasing



(a) Accuracy vs. efficiency

(b) Performance w.r.t. MBR size

Figure 5.12: Heuristics for partial domination

the number of allowed $DDC_{Optimal}$ calls. In particular, the accuracy increases very fast at the beginning of the partitioning process but slows down later on. We can also observe that DC_{bisect} significantly outperforms the other approaches when allowing more than 27 $DDC_{Optimal}$ calls per MBR, while DC_{grid} performs best for 8 or less $DDC_{Optimal}$ calls.

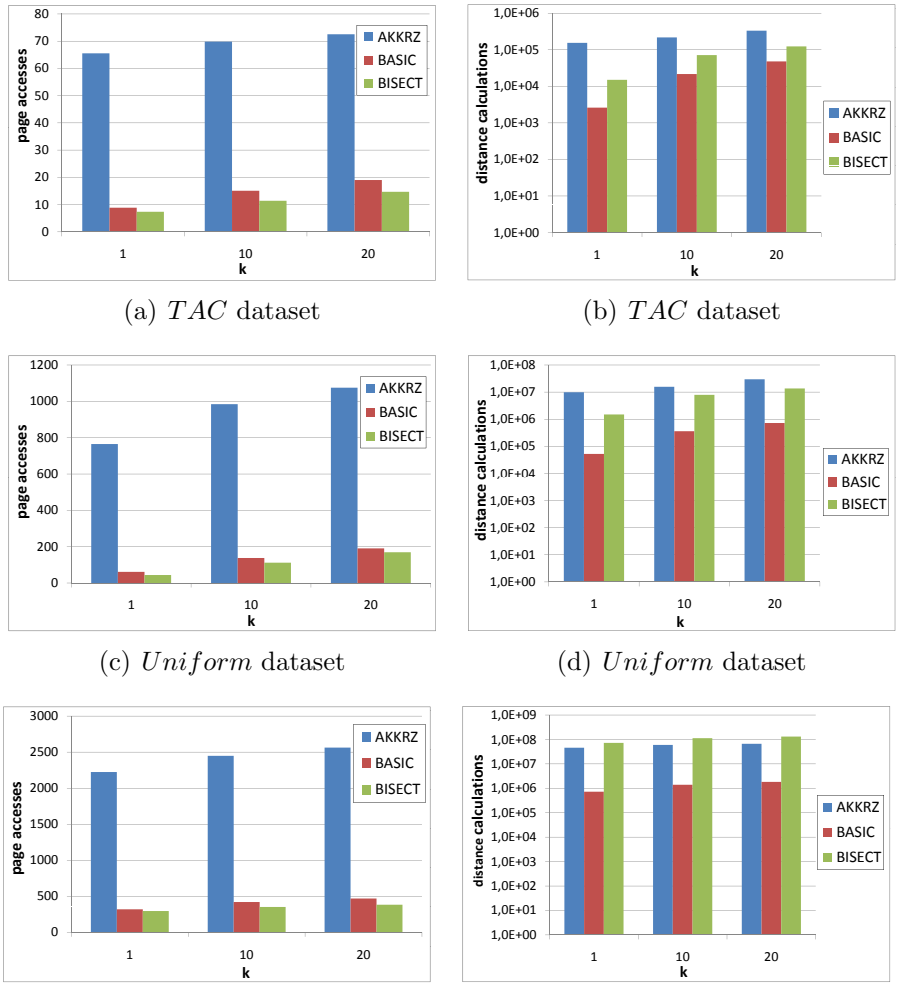
In the next experiment, as shown in Figure 5.12(b), we fixed the number $DDC_{Optimal}$ calls per MBR to 64 and varied their extent. We measured the gain of the domination count over DC_{basic}. Here, again, DC_{bisect} outperforms the other approaches in particular for larger MBR sizes. Note that for a given application, the optimal number of partitions depends on the cost for evaluating a candidate object. The higher that cost, the more partitions can be used in order to reduce the total runtime.

5.7.3 Impact on Standard Spatial Query Processing Methods

In our last experiments, we evaluate the impact of our approaches on the performance of standard query processing methods. Here, we refer to Section 5.6, describing how our methods can be plugged into state-of-the-art query processing methods. In particular, we exemplarily consider the most prominent query methods, the k-nearest neighbor (k-NN) search and the reverse k-nearest neighbor (Rk-NN) search. For this evaluation we use one synthetic dataset, containing 100k uniformly distributed 5D points, and two real world datasets *TAC*[210] consisting of 705099 2D points and *Forest*[86] containing 581012 10D points.

First, we evaluate the impact of our two domination-count estimation approaches DC_{basic} and DC_{bisect} on a reverse k-nearest neighbor search method. As a baseline, we use the algorithm proposed in [4] (in the following referred to *AKKRZ*) for Rk-NN search on the Euclidean space using an R^*-Tree[1]. The *AKKRZ* algorithm originally uses the Min-/MaxDist decision criterion to conservatively prune candidates. We evaluate the impact by comparing the query performance of the original AKKRZ algorithm with the version

[1]We use the R^*-Tree provided in the Elki Framework [3]



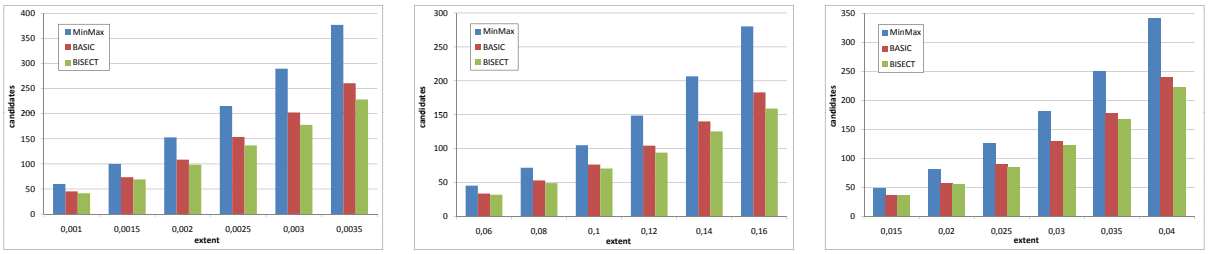
(a) *TAC* dataset (b) *TAC* dataset

(c) *Uniform* dataset (d) *Uniform* dataset

(e) *Forest* dataset (f) *Forest* dataset

Figure 5.13: *AKKRZ* using different decision criteria. Page accesses (left side) and distance calculations (right side).

where we replace the domination count estimation with our methods. Note, that with except of the domination count estimation method, both Rk-NN versions are identical. The results illustrated in Figures 5.13(a), 5.13(c) and 5.13(e) show the query performance of both Rk-NN versions in terms of average number of page accesses for varying parameter k and different datasets. It is notable that the enhanced algorithm requires less page access by almost a full order of magnitude on all datasets. Using DC_{bisect} to apply the paradigm of partial pruning based on bisections (c.f. Section 5.5.2) with a maximum number of ten splits per MBR, the number of page accesses can be significantly dropped even further. The large performance increase compared to the original version of *AKKRZ* can be explained by the fact that our domination decision criterion has a much higher pruning power on large MBRs compared to the original version that is based on the Min/MaxDist criterion. This allows us to prune candidates already on a high directory level and, thus, to prune a



(a) *TAC* dataset (2D) (b) *Uniform* dataset (5D) (c) *Forest* dataset (10D)

Figure 5.14: Evaluation of the different decision criteria for 10 nearest neighbor queries.

large number of candidate MBRs very early.

Beside the I/O cost, it is also important to consider the cpu cost since the accuracy of our domination count estimation methods is highly influenced by the cpu cost spent for the estimation process, as shown in the previous section. For this reason, in addition to the I/O cost evaluation we evaluate the cpu cost measured by the number of distance calculations required for the competing techniques as the cpu cost are mainly distance computation bounded. We counted the total number of distance calculations. Calls of $DDC_{Optimal}$ and $DDC_{\mathbf{MinMax}}$ were penalized with two distance calculations[2]. The resulting numbers of total distance calculations are shown in Figures 5.13(b), 5.13(d) and 5.13(f). It can be observed that the enhanced *AKKRZ* algorithm using DC_{basic} significantly outperforms the basic *AKKRZ* by close to two orders of magnitude. The rationale for this is that the number of calculations increases quadratic in the number of candidates. However, the high computational cost required when applying partial pruning becomes evident here. Using DC_{bisect} with a maximum of ten splits, the number of distance calculations increases by a factor of about five.

Finally, we evaluate the impact of $DDC_{Optimal}$ and partial domination on k-NN queries among objects approximated by MBRs. These experiments are based on three artificial datasets that rely on the three datasets used in the foregoing experiments (TAC, Uniform, Forest). Each vector in a dataset defines the center of an MBR. For each of the resulting datasets 100 MBRs were chosen randomly as query MBR Q for a 10-NN query on the remaining dataset. Here we did not apply any index structure. The performance of the competing approaches were measured by the average number of candidates that could neither be pruned nor be reported as true hits. The results showing the performance in terms of the number of remaining candidates are depicted in Figure 5.14 for varying extent of the MBRs. It can be observed, that DC_{basic} significantly reduces the number of candidates compared to pruning based on $DDC_{\mathbf{MinMax}}$ on all datasets. The relative performance boost increases for an increasing extent of the MBRs. We also found out in our experiments, that the parameter k has no significant influence on the relative performance boost. Figure 5.14 also shows, that DC_{bisect} is able to further boost the performance, especially for large MBRs.

[2]in concordance with run-time experiments omitted here due to space considerations

5.8 Conclusions

The concept of domination is a very useful tool to perform spatial pruning on rectangles in a wide field of applications. Current state-of-the-art approaches are either incomplete or scale exponentially in the number of dimensions. In this chapter we proposed a decision criterion that is complete and efficiently computable in $O(d)$. In addition, we discuss how this decision criterion can be used to accurately estimate the domination count of objects by incorporating information about partial domination. While all current approaches that use information about partial domination can only be used on two dimensional data our solution can be applied to data of arbitrary dimensionality. Our experimental evaluation shows that our novel decision criterion can be used to vastly increase the pruning power of existing applications by several orders of magnitude. For future work, we plan to plug-in our novel decision criterion to more existing applications. In addition we will explore decision criteria for object representations having non-rectangular shape.

Chapter 6

Probabilistic k-Nearest Neighbor Queries on Uncertain Data

In this section, an effective and efficient probabilistic pruning criterion for k-Nearest Neighbor Queries queries on uncertain data is presented. This criterion allows to assess the probability that an uncertain object A is closer to an uncertain query object Q than a third uncertain object B. This approach supports a general uncertainty model using continuous probabilistic density functions to describe the (possibly correlated) uncertain attributes of objects. In a nutshell, the problem to be solved is to compute the PDF of the random variable denoted by the *probabilistic domination count*: Given an uncertain database object B, an uncertain query object Q and a set $\mathcal{DB}$ of uncertain database objects in a multi-dimensional space, the probabilistic domination count denotes the number of uncertain objects in $\mathcal{DB}$ that are closer to Q than B. Unlike the domination count introduced in Definition 26 in Chapter 5, the probabilistic domination count is defined on uncertain data, and thus, is not a single scalar, but a random variable. This probabilistic domination count is used to efficiently answer probabilistic k-Nearest Neighbor queries on uncertain data using possible world semantics. Based on the spatial pruning criterion presented in Chapter 5 a novel geometric pruning filter is proposed and an iterative filter-refinement strategy for conservatively and progressively estimating the probabilistic domination count is introduced. An experimental evaluation shows that the proposed technique allows to acquire tight probability bounds for the probabilistic domination count quickly, even for large uncertain databases.

6.1 Introduction

This section introduce a novel scalable pruning approach to identify candidates for a class of probabilistic similarity queries. This novel pruning method is applied to the most prominent query of the above mentioned class, the probabilistic k-nearest neighbor (PkNN) query.

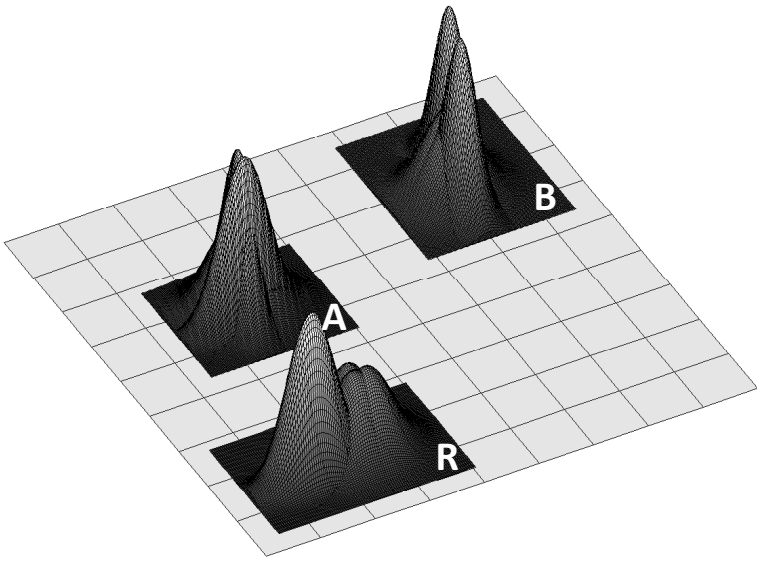


Figure 6.1: A dominates B w.r.t. R with high probability.

6.1.1 Uncertainty Model

In this chapter, we assume a continuous model, where each object $U_i \in \mathcal{DB} = \{U_1, ..., U_N\}$ is represented by a continuous probability density function f_i. Following the convention of uncertain databases [31, 45, 48, 51, 55, 131, 170], we assume that f_i is (minimally) bounded by an *uncertainty region* $U_i^{\square}$ such that $\forall x \notin U_i^{\square} : f_i(x) = 0$ and

$$\int_{U_i^{\square}} f_i(x)dx \leq 1.$$

Specifically, the case $\int_{U_i^{\square}} f_i(x)dx < 1$ implements existential uncertainty, i.e. object o_i may not exist in the database at all with a probability greater than zero. In this chapter we focus on the case $\int_{U_i^{\square}} f_i(x)dx = 1$, but the proposed concepts can be easily adapted to existentially uncertain objects. If f_i is an unbounded PDF, e.g., Gaussian PDF, we truncate PDF tails with negligible probabilities and normalize the resulting PDF. This procedure is also used in related work [45, 48, 31]. In specific, [31] shows that for a reasonable low truncation threshold, the impact on the accuracy of probabilistic ranking queries is very low.

In this way, each uncertain object can be considered as a d-dimensional rectangle with an associated multi-dimensional object PDF (c.f. Figure 6.1). Here, we assume that uncertain attributes may be mutually dependent. Therefore the object PDF can have any arbitrary form, and in general, cannot simply be derived from the marginal distribution of the uncertain attributes. Note that in many applications, a discrete uncertainty model is appropriate, meaning that the probability distribution of an uncertain object is given by a finite number of alternatives assigned with probabilities. This is a special case of the model used here.

6.1.2 Problem Formulation

To answer kNN queries on uncertain objects efficiently, we exploit the observation that an object o is a k-nearest neighbor of an object q if and only if there is less than k objects in the database, that are closer to q than o. Thus, we address the problem of detecting for a given uncertain query object Q and an uncertain object B the number of uncertain objects of an uncertain database $\mathcal{DB}$ that are closer to (*i.e. dominate*) Q than B. We call this number the *domination count* of B w.r.t. Q as defined in the following.

Definition 30 (Probabilistic Domination). *Consider an uncertain database $\mathcal{DB} = \{U_1, ..., U_N\}$ and an uncertain query object Q. Let $A, B \in \mathcal{DB}$. Let $(A \prec_Q B)$ denote the random indicator variable that returns one, if and only if A dominates B w.r.t. Q, formally:*

$$(A \prec_Q B) := \mathcal{I}(dist(A,Q) < dist(B,Q), \mathcal{DB})$$

where $\mathcal{I}(dist(A,Q) < dist(B,Q), \mathcal{DB})$ is a random indicator variable that returns one if the random location of A is closer to the random location of Q than the random location of B, and zero otherwise.

Note that in Section 5, the predicate $(A \prec_Q B)$ has been defined for rectangles. In the above Definition 30, the notation $(A \prec_Q B)$ has deliberately been overloaded as a random variable having uncertain objects as formal parameters, rather than a predicate having rectangles as formal parameters. The random variable $(A \prec_Q B)$ follows a Bernoulli distribution, i.e., a distribution having a success probability $P(A \prec_Q B)$ of taking value one, and a $1 - P(A \prec_Q B)$ probability of taking value zero.

Definition 31 (Domination Probability). *The probability $P(A \prec_Q B)$ that object A dominates object B with respect to Q is denoted as* domination probability. *If $P(A \prec_Q B) = 0$, then we say that A does not dominate B with respect to Q. If $P(A \prec_Q B) = 1$, then we say that A certainly dominates B with respect to Q. If $0 < P(A \prec_Q B) < 1$, then we stay that A dominates B probabilistically with respect to Q. This case is called* probabilistic domination.

Definition 32 (Probabilistic Domination Count). *Consider an uncertain database $\mathcal{DB} = \{U_1, ..., U_N\}$ and an uncertain query object Q. For each uncertain object $B \in \mathcal{DB}$, the probabilistic domination count $DomCount(B,Q)$ is defined as the random variable of the number of uncertain objects $A \in \mathcal{DB}$ $(A \neq B)$ that are closer to q than B:*

$$DomCount(B,Q) := \sum_{A \in \mathcal{DB}, A \neq B} (A \prec_Q B)$$

$DomCount(B,Q)$ is the sum of $N-1$ non-necessarily identically distributed and non-necessarily independent Bernoulli variables. The domination count can be used directly to efficiently answer probabilistic threshold k-NN queries.

Corollary 5. *Let Q be an uncertain query object and let k be a scalar. The problem is to find all uncertain objects $kNN_\tau(Q)$ that are the k-nearest neighbors of Q with a probability of at least τ. Given the probability mass function of $DomCount(B,Q)$, we can compute the probability $P^{kNN}(B,Q)$ that an object B is a kNN of Q as follows:*

$$P^{kNN}(B,Q) = \sum_{i=0}^{k-1} P(DomCount(B,Q) = i)$$

Proof. The above corollary is evident, since the proposition "B is a kNN of Q" is equivalent to the proposition "B is dominated by less than k objects". □

To decide whether B is a kNN of Q, i.e. whether $B \in P\tau\, kNN(Q, \mathcal{DB})$, we just need to check if $P^{kNN}(B,Q) \geq \tau$.

The problem solved in this chapter is to efficiently compute the probability mass function of $DomCount(B,Q)$. The solutions to compute the sum of independent Bernoulli trials presented in Section 3.3 cannot applied directly, since the two random events $(A_i \prec_Q B)$ and $(A_j \prec_Q B)$ are mutually dependent, as they both depend on the location of uncertain objects B and Q. In this chapter, the technique of generating functions, introduced in Section 3.3 will be adapted, to give upper and lower bound functions of the probability mass function of $DomCount(B,Q)$. Experiments will show, that the resulting bound functions are very tight, allowing to guarantee correct kNN results in most cases, avoiding expensive integration to obtain exact result.

6.1.3 Basic Idea

First (Section 6.3) presents a methodology to efficiently find objects in $\mathcal{DB}$ that certainly dominate B w.r.t. Q as well as objects in $\mathcal{DB}$ that do not dominate B. At the same time, we find the set of objects that dominate B probabilistically. Using a decomposition technique, for each object A in this set, we can derive a lower and an upper bound for $P(A \prec_Q B)$, i.e., for the probability that A dominates B w.r.t. Q. In Section 6.4, we show that due to dependencies between object distances to Q, these probabilities cannot be combined in a straightforward manner to approximate the distribution of $DomCount(B,Q)$. We propose a solution that copes with these dependencies and introduce techniques that help to to compute the probabilistic domination count in an efficient way. In particular, we prove that the bounds of $P(A \prec_Q B)$ are mutually independent if they are computed without a decomposition of B and Q. Then, we provide a class of uncertain generating functions that use these bounds to bound the distribution of $DomCount(B,Q)$. We then propose an algorithm which progressively refines $DomCount(B,Q)$ by iteratively decomposing the objects that influence its computation in Section 6.5. In Section 6.6, we experimentally demonstrate the effectiveness and efficiency of our probabilistic pruning methods for various parameter settings on artificial and real-world datasets.

6.2 Related Work

Uncertain similarity query processing has focused on various aspects. A lot of existing work dealing with uncertain data addresses probabilistic one nearest neighbor (1NN) queries for certain query objects [51, 110] and for uncertain queries [95]. To reduce computational effort, [48] add threshold constraints in order to retrieve only objects whose probability of being the nearest neighbor exceeds a user-specified threshold to control the desired confidence required in a query answer. Similar semantics of queries in probabilistic databases are provided by Top-k nearest neighbor queries [31], where the k most probable results of being the nearest neighbor to a certain query point are returned. The solution of [133] for probabilistic k-nearest neighbor (kNN) queries is restricted to expected distances of uncertain objects to the query object. The use of expected distances drops any uncertainty information, yielding expected results, whose reliability cannot be assessed ([171, 125]).

The work of [49] used result based query semantics to answer probabilistic kNN queries. For a probabilistic kNN query, the set of possible answers using result based semantics equals $\mathcal{DB}^k$, i.e., one possible result for each k-combination of objects in $\mathcal{DB}$. Since in the worst case, the result is exponentially in this case, the problem itself must be exponentially hard, since returning the computed results requires exponential time. For this reason, [49] presents approximation techniques, to find result sets having a probability of at least τ. However, due to the exponential large set of possible answers, a proper τ value that still returns any results, becomes exponentially small. Furthermore, the approach of [49] assumes a certain query point.

Several approaches return the full result to queries as a ranking of probabilistic objects according to their distance to a certain query point [27, 55, 125, 171]. However, all these prior works have in common that the query is given as a single (certain) point.

6.3 Similarity Domination on Uncertain Data

In this section, we tackle the following problem: Given three uncertain objects A, B and Q in a multidimensional space $\mathbb{R}^d$, determine whether object A is closer to Q than B w.r.t. a distance function defined on the objects in $\mathbb{R}^d$. If this is the case, we say A *dominates* B w.r.t. Q. In contrast to Chapter 5, where this problem is solved for certain data, in the context of uncertain objects this domination relation is not a predicate that is either true or false, but rather a (dichotomous) random variable as defined in Definition 30. In the example depicted in Figure 6.1, there are three uncertain objects A, B and R, each bounded by a rectangle representing the possible locations of the object in $\mathbb{R}^2$. The PDFs of A, B and R are depicted as well. In this scenario, we cannot determine for sure whether object A dominates B w.r.t. R. However, it is possible to determine that object A dominates object B w.r.t. R with a high probability. If $Q = R$ is the uncertain query object of a probabilistic 1NN queries, we can guarantee that B must have a low probability to be a result of this query, because A has a high probability to be closer to Q than B.

The problem at issue is to determine the *domination probability* $P(A \prec_Q B)$ as defined in Definition 31. Naively, we can compute $P(A \prec_Q B)$ by simply integrating the probability of all possible worlds in which A dominates B w.r.t. Q exploiting inter-object independency:

$$P(A \prec_Q B) = \int_{a \in A} \int_{b \in B} \int_{q \in Q} \mathcal{I}(a,b,q) \cdot P(A=a) \cdot P(B=b) \cdot P(Q=q) da\, db\, dq,$$

where $\mathcal{I}(a,b,q)$ is the following (crisp) indicator function:

$$\mathcal{I}(a,b,r) = \begin{cases} 1, & \text{if } dist(a,r) < dist(b,r) \\ 0, & \text{else} \end{cases}$$

The problem of this naive approach is the computational cost of the triple-integral. The integrals of the PDFs of A, B and Q may in general not be representable as a closed-form expression and the integral of $\mathcal{I}(a,b,q)$ does not have a closed-from expression. Therefore, an expensive numeric approximation is required for this approach. In the rest of this section we propose methods that efficiently derive bounds for $P(A \prec_Q B)$, which can be used to prune objects, thus avoiding integral computations.

6.3.1 Complete Domination

First, we show how to detect whether A completely dominates B w.r.t. Q (i.e. if $P(A \prec_Q B) = 1$) based only on the rectangular approximations $A^\Box$, $B^\Box$ and $Q^\Box$. The state-of-the-art criterion to detect spatial domination on rectangular uncertainty regions is with the use of minimum/maximum distance approximations. This criterion states that A dominates B w.r.t. Q if the minimum distance between $Q^\Box$ and $B^\Box$ is greater than the maximum distance between $Q^\Box$ and $A^\Box$. Although correct, this criterion is not tight (cf. Chapter 5), i.e. not each case where $A^\Box$ dominates $B^\Box$ w.r.t. $Q^\Box$ is detected by the min/max-domination criterion. The problem is that the dependency between the two distances between A and Q and between B and Q is ignored. Obviously, the distance between A and Q as well as the distance between B and Q depend on the location of Q. However, since Q can only have a unique location within its uncertainty region, both distances are mutually dependent. To obtain a tighter decision criterion, we adopt the spatial domination concepts proposed in Chapter 5 for rectangular uncertainty regions.

Corollary 6 (Complete Domination)**.** *Let A, B, Q be uncertain objects having rectangular space approximation $A^\Box$, $B^\Box$ and $Q^\Box$, respectively. The following implication holds:*

$$P(A \prec_Q B) = 1 \Leftarrow \tag{6.1}$$
$$\sum_{i=1}^{d} \max_{q_i \in \{Q_i^{min}, Q_i^{max}\}} (MaxDist(A_i, q_i)^p - MinDist(B_i, q_i)^p) < 0,$$

where A_i, B_i and Q_i denote the projection interval of the respective rectangular uncertainty region $A^\Box$, $B^\Box$ and $Q^\Box$ on the i^{th} dimension; Q_i^{min} (Q_i^{max}) denotes the lower (upper) bound of interval Q_i, and p corresponds to the used L_p norm. The functions $MaxDist(A_i, q_i)$ and $MinDist(A_i, q_i)$ denote the maximal (respectively minimal) distance between the one-dimensional interval A_i and the one-dimensional point q_i.

Proof. In Section 5.4 of Chapter 5 it is shown that the right hand side of implication 6.1 is equivalent to the following statement:

$$\forall a \in A^\Box, b \in B^\Box, q \in Q^\Box : dist(a, b) < dist(q, b) \tag{6.2}$$

which is true if and only if for each $a \in A^\Box, b \in B^\Box, q \in Q^\Box$ it holds that a is closer to b than q, where $A^\Box, B^\Box$ and $Q^\Box$ are rectangular approximations. By definition of the possible worlds model, the set of combinations $a \in A^\Box, b \in B^\Box, q \in Q^\Box$ corresponds to a superset of all possible worlds. Consequently, the above Predicate 6.2 implies that $\forall a \in A, b \in B, q \in Q : \mathcal{I}(dist(a, b) < dist(q, b)) = 1$ by definition of indicator function $\mathcal{I}$. Using Equation 8.1 we obtain the triple-sum

$$P(A \prec_Q B) =$$
$$\sum_{a_i \in A} \sum_{b_j \in B} \sum_{q_k \in Q} 1 \cdot P(a_i) \cdot P(b_j) \cdot P(q_k)$$

As we can see, the above triple-sum is equal to the sum of the probabilities of *all* possible worlds which is equal to one. Consequently, we obtain $P(A \prec_Q B) = 1$. □

In addition, it holds that

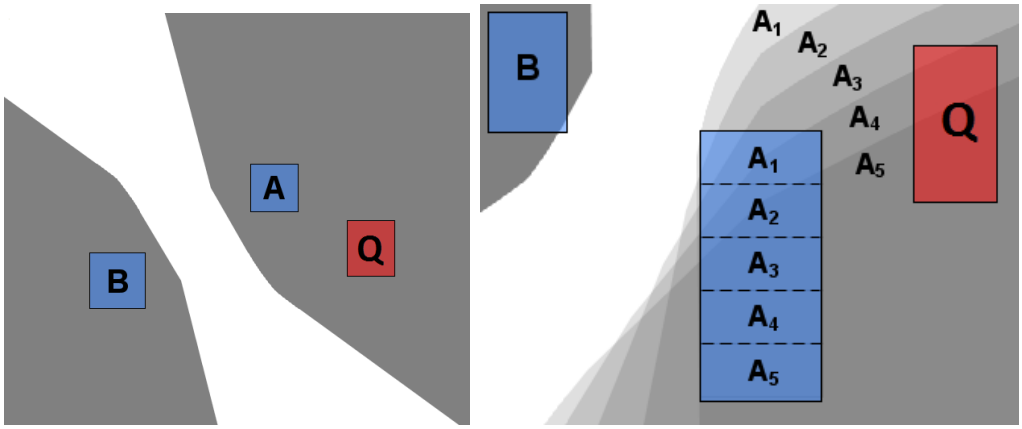
Corollary 7.

$$P(A \prec_Q B) = 1 \Leftrightarrow P(B \prec_Q A) = 0$$

Proof. The above corollary is evident, since in any world where $A = a$, $B = b$ and $Q = q$ it holds that $(A \prec_Q B) = 1 - (B \prec_Q A)$. Thus, if and only if $(A \prec_Q B) = 1$ in all possible worlds, then $(B \prec_Q A) = 0$ in all possible worlds and vice versa. □

Lemma 16 (Complete Non-Domination)**.** *Let A, B, Q be uncertain objects having rectangular space approximation $A^\Box$, $B^\Box$ and $Q^\Box$, respectively. The following statement holds:*

$$P(B \prec_Q A) = 0 \Leftarrow$$
$$\sum_{i=1}^{d} \max_{q_i \in \{Q_i^{min}, Q_i^{max}\}} (MaxDist(A_i, q_i)^p - MinDist(B_i, q_i)^p) < 0,$$



(a) Complete domination (b) Probabilistic domination

Figure 6.2: Similarity Domination.

Proof. This Lemma follows directly from Lemma 6 and Corollary 7. □

In the example depicted in Figure 6.2(a), the grey region on the right shows all points that definitely are closer to A than to B and the grey region on the left shows all points that definitely are closer to B than to A. Consequently, A dominates B (B dominates A) if Q completely falls into the right (left) grey shaded half-space.[1]

6.3.2 Probabilistic Domination

Now, we consider the case where A does not completely dominate B w.r.t. Q. In consideration of the possible world semantics, there may exist worlds in which A dominates B w.r.t. Q, but not all possible worlds may satisfy this criterion. Let us consider the example shown in Figure 6.2(b) where the uncertainty region of A is decomposed into five partitions, each assigned to one of the five grey-shaded regions illustrating which points are closer to the partition in A than to B. As we can see, Q only completely falls into three grey-shaded regions. This means that A does not completely dominate B w.r.t. Q. However, we know that in some possible worlds (at least in all possible words where A is located in A_1, A_2 or A_3) A does dominate B w.r.t. Q. The question at issue is how to determine the probability $P(A \prec_Q B)$ that A dominates B w.r.t. Q in an efficient way. The key idea is to decompose the uncertainty region of an object X into subregions for which we know the probability that X is located in that subregion (as done for object A in our example). Therefore, if neither $(A \prec_Q B)$ nor $(B \prec_Q A)$ holds, then there may still exist subregions $A' \subset A$, $B' \subset B$ and $Q' \subset Q$ such that $(A' \prec_{Q'} B')$ holds. Given disjunctive rectangular decomposition schemes $\underline{\mathcal{A}}$, $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ we can identify triples of subregions ($A' \in \underline{\mathcal{A}}$, $B' \in \underline{\mathcal{B}}$, $Q' \in \underline{\mathcal{Q}}$) for which $(A' \prec_{Q'} B')$ holds. Let $\mathcal{I}(A', B', Q')$ be the following indicator function:

[1]Note that the grey regions are not explicitly computed; we only include them in Figure 6.2(a) for illustration purpose.

$$\mathcal{I}(A', B', Q') = \begin{cases} 1, & \text{if } (A' \prec_{Q'} B') \\ 0, & \text{else} \end{cases}$$

Lemma 17. *Let A, B and Q be uncertain objects with disjunctive rectangular object decompositions $\underline{\mathcal{A}}, \underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$, respectively. To derive a lower bound $P_{LB}(A \prec_Q B)$ of the probability $P(A \prec_Q B)$ that A dominates B w.r.t. Q, we can accumulate the probabilities of combinations of these subregions as follows:*

$$P_{LB}(A \prec_Q B) =$$

$$\sum_{A' \in \underline{\mathcal{A}}, B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P(a \in A') \cdot P(b \in B') \cdot P(r \in Q') \cdot \mathcal{I}(A', B', Q'),$$

where $P(X \in X')$ denotes the probability that object X is located within the region X'.

Proof. The probability of a combination (A', B', Q') can be computed by $P(a \in A') \cdot P(b \in B') \cdot P(r \in Q')$ due to the assumption of mutually independent objects. These probabilities can be aggregated due to the assumption of disjunctive subregions, which implies that any two different combinations of subregions $(A' \in \underline{\mathcal{A}}, B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}})$ and $(A'' \in \underline{\mathcal{A}}, B'' \in \underline{\mathcal{B}}, Q'' \in \underline{\mathcal{Q}}, A' \neq A'' \vee B' \neq B'' \vee Q' \neq Q''$ must represent disjunctive sets of possible worlds. It is obvious that all possible worlds defined by combinations (A', B', Q') where $\mathcal{I}(A', B', Q') = 1$, A dominates B w.r.t. Q. But not all possible worlds where A dominates B w.r.t. Q are covered by these combinations and, thus, do not contribute to $P_{LB}(A \prec_Q B)$. Consequently, $P_{LB}(A \prec_Q B)$ lower bounds $P(A \prec_Q B)$. □

Analogously, we can define an upper bound of $P(A \prec_Q B)$:

Lemma 18. *An upper bound $P_{UB}(A \prec_Q B)$ of $P(A \prec_Q B)$ can be derived as follows:*

$$P_{UB}(A \prec_Q B) = 1 - P_{LB}(B \prec_Q A)$$

Naturally, the more refined the decompositions are, the tighter the bounds that can be computed and the higher the corresponding cost of deriving them. In particular, starting from the entire MBRs of the objects, we can progressively partition them to iteratively derive tighter bounds for their dependency relationships until a desired degree of certainty is achieved (based on some threshold). However, in the next section, we show that the derivation of the domination count $DomCount(B, Q)$ of a given object B (cf. Definition 32), which is the main module of prominent probabilistic queries cannot be straightforwardly derived with the use of these bounds and we propose a methodology based on generating functions for this purpose.

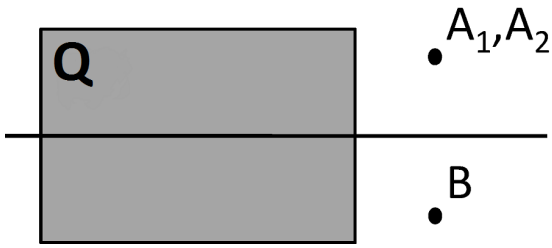


Figure 6.3: A_1 and A_2 dominate B w.r.t. Q with a probability of 50%, respectively.

6.4 Probabilistic Domination Count

In Section 6.3 we described how to conservatively and progressively approximate the probability that A dominates B w.r.t. Q. Given these approximations $P_{LB}(A \prec_Q B)$ and $P_{UB}(A \prec_Q B)$, the next problem is to cumulate these probabilities to get an approximation of the domination count $DomCount(B, Q)$ of an object B w.r.t. Q (cf. Definition 32). To give an intuition how challenging this problem is, we first present a naive solution that can yield incorrect results due to ignoring dependencies between domination relations in Section 6.4.1. To avoid the problem of dependent domination relations, we first show in Section 6.4.2 how to exploit object independencies to derive domination bounds that are mutually independent. Afterwards, in Section 6.4.3, we introduce a new class of uncertain generating functions that can be used to derive bounds for the domination count efficiently, as we show in Section 6.4.4. To motivate this new class of generating functions, Section 6.4.5 shows that the traditional generating function technique fails to compute tight probability bounds. Finally, in Section 6.4.6, we show how to improve our domination count approximation by considering disjunct subsets of possible worlds for which a more accurate approximation can be computed.

6.4.1 The Problem of Domination Dependencies

To compute $DomCount(B, Q)$, a straightforward solution is to first approximate $P(A \prec_Q B)$ for all $A \in \mathcal{DB}$ using the technique proposed in Section 6.3. Then, given these probabilities we can apply the technique of uncertain generating functions (cf. Section 6.4.3) to approximate the probability that exactly 0, exactly 1, ..., exactly $n-1$ uncertain objects dominate B. However, this approach ignores possible dependencies between domination relationships. Although we assume independence between objects, the random variables $(A_1 \prec_Q B)$ and $(A_2 \prec_Q B)$ are mutually dependent because the distance between A_1 and Q depends on the distance between A_2 and Q because object Q can only appear once. Consider the following example:

Example 17. *Consider a database of three certain objects B, A_1 and A_2 and the uncertain query object Q, as shown in Figure 6.3. For simplicity, objects A_1 and A_2 have the same position in this example. The task is to determine the domination count of B w.r.t. Q. The domination half-space for A_1 and A_2 is depicted here as well. Let us assume that A_1*

(A_2) dominates B with a probability of $P(A_1 \prec_Q B) = P(A_2 \prec_Q B) = 50\%$. Recall that this probability can be computed by integration or approximated with arbitrary precision using the technique of Section 6.3. However, in this example, the probability that both A_1 and A_2 dominate B is not simply $50\% \cdot 50\% = 25\%$, as the generating function technique would return.

The reason for the wrong result in this example, is that the generating function requires mutually independent random variables. However, in this example, it holds that if and only if Q falls into the domination half-space of A_1, it also falls into the domination half-space of A_2. Thus we have the dependency $(A_1 \prec_Q B) \leftrightarrow (A_2 \prec_Q B)$ and the probability for Q to be dominated by both A_1 and A_2 is

$$P(A_1 \prec_Q B) \cdot P(A_2 \prec_Q B) | (A_1 \prec_Q B))$$
$$= 0.5 \cdot 1 = 0.5.$$

This example has shown that ignoring the dependency between two domination random variables $(A_1 \prec_Q B)$ and $(A_1 \prec_Q B)$ can significantly distort the probabilistic domination count.

6.4.2 Domination Approximations Based on Independent Objects

In general, domination relations may have arbitrary correlations. Therefore, we present a way to compute the domination count $DomCount(B, Q)$ while accounting for the dependencies between domination relations.

Complete Domination

In an initial step, *complete domination* serves as a filter which allows us to detect those objects $A \in \mathcal{DB}$ that definitely dominate a specific object B w.r.t. Q and those objects that definitely do not dominate B w.r.t. Q by means of evaluating $(A^\Box \prec_{Q^\Box} B^\Box)$ using the spatial domination decision criterion of Chapter 5. It is important to note that complete domination relations are mutually independent, since complete domination is evaluated on the entire uncertainty regions of the objects. After applying complete domination, we have detected objects that dominate B in all, or no possible worlds. Consequently, we get a first approximation of the domination count $DomCount(B, Q)$, obviously, it must be higher than the number N of objects that dominate B and lower than $|\mathcal{DB}| - M$, where M is the number of objects that dominate B in no possible world, i.e. $P(DomCount(B, Q) = k) = 0$ for $k \leq N$ and $k \geq |\mathcal{DB}| - M$. Nevertheless, for $N < k < |\mathcal{DB} - M|$ we still have a very bad approximation of the domination count probability of $0 \leq P(DomCount(B, Q) = k) \leq 1$.

Probabilistic Domination

In order to refine this probability distribution, we have to take the set of *influence* objects $influenceObjects = \{A_1, ..., A_C\}$, which neither completely prune B nor are completely dominated by B w.r.t. Q. For each $A_i \in influenceObjects$, $0 < P(A_i \prec_Q B) < 1$. For these objects, we can compute probabilities $P(A_1 \prec_Q B), ..., P(A_C \prec_Q B)$ according to the methodology in Section 6.3. However, due to the mutual dependencies between domination relations (cf. Section 6.4.1), we cannot simply use these probabilities directly, as they may produce incorrect results. However, we can use the observation that the objects A_i are mutually independent and each candidate object A_i only appears in a single domination relation $(A_1 \prec_Q B), ..., (A_C \prec_Q B)$. Exploiting this observation, we can decompose the objects $A_1, ..., A_C$ only, to obtain mutually independent bounds for the probabilities $P(A_1 \prec_Q B), ..., P(A_C \prec_Q B)$, as stated by the following lemma:

Lemma 19. *Let $A_1, ...A_C$ be uncertain objects with disjunctive object decompositions $\mathcal{A}_1, ..., \mathcal{A}_C$, respectively. Also, let B and Q be uncertain objects (without any decomposition). The lower (upper) bound $P_{LB}(A_i \prec_Q B)$ ($P_{UB}(A_i \prec_Q B)$) as defined in Lemma 17 (Lemma 18) of the random variable $(A_i \prec_Q B)$ is independent of the random variable $(A_j \prec_Q B)$ $(1 \leq i \neq j \leq C)$.*

Proof. Consider the random variable $(A_i \prec_Q B)$ conditioned on the event $(A_j \prec_Q B) = 1$. Using Equation 17, we can derive the lower bound probability of $(A_i \prec_Q B) = 1 | (A_j \prec_Q B) = 1$ by conditioning all random variables as follows:

$$P_{LB}(A_i \prec_Q B | (A_j \prec_Q B) = 1) =$$

$$\sum_{A'_i \in \mathcal{A}_i, B' \in \mathcal{B}, Q' \in \mathcal{Q}} [P(a_i \in A'_i | (A_j \prec_Q B) = 1) \cdot$$

$$P(b \in B' | (A_j \prec_Q B) = 1) \cdot$$

$$P(r \in Q' | (A_j \prec_Q B) = 1) \cdot \mathcal{I}(A'_i, B', Q')]$$

Note that the indicator function $\mathcal{I}(A'_i, B', Q')$ is not a random variable, and thus is not conditioned.[2] Now we exploit that B and Q are not decomposed, thus $B' = B$ and $Q' = Q$, and thus $P(B \in B' | (A_j \prec_Q B) = 1) = 1 = P(B \in B')$ and $P(Q \in Q' | (A_j \prec_Q B) = 1) = 1 = P(Q \in Q')$. We obtain:

$$P_{LB}(A_i \prec_Q B | (A_j \prec_Q B) = 1) =$$

$$\sum_{A'_i \in \mathcal{A}_i, B' \in \mathcal{B}, Q' \in \mathcal{Q}} [P(a_i \in A'_i | (A_j \prec_Q B) = 1) \cdot$$

$$P(b \in B') \cdot P(r \in Q') \cdot \mathcal{I}(A'_i, B', Q')]$$

[2] Recall that $\mathcal{I}(A'_i, B', Q')$ returns one only if A'_i spatially dominates B' with respect to Q'. This spatial (full) domination is independent of any random events.

Next we exploit that $P(a_i \in A_i' | (A_j \prec_Q B) = 1) = P(a_i \in A_i')$ since A_i is independent of the random variable $(A_j \prec_Q B)$ and obtain:

$$P_{LB}(A_i \prec_Q B | (A_j \prec_Q B) = 1) =$$

$$\sum_{A_i' \in \mathcal{A}_i, B' \in \mathcal{B}, Q' \in \mathcal{Q}} [P(a_i \in A_i') \cdot P(b \in B') \cdot P(r \in Q') \cdot \mathcal{I}(A_i', B', Q')]$$

$$= P_{LB}(A_i \prec_Q B)$$

Analogously, it can be shown that

$$P_{UB}(A_i \prec_Q B | (A_j \prec_Q B) = 1) = P_{UB}(A_i \prec_Q B).$$

□

In summary, we can now derive, for each object A_i a lower and an upper bound of the probability that A_i dominates B w.r.t. Q. However, these bounds may still be rather loose, since we only consider the full uncertainty region of B and Q so far, without any decomposition. In Section 6.4.6, we will show how to obtain more accurate, still mutual independent probability bounds based on decompositions of B and Q. Due to the mutual independency of the lower and upper probability bounds, these probabilities can now be used to get an approximation of the domination count of B. In order to do this efficiently, we adapt the generating functions technique which is proposed in [125]. The main challenge here is to extend the generating function technique in order to cope with probability bounds instead of concrete probability values. It can be shown that a straightforward solution based on the existing generating functions technique applied to the lower/upper probability bounds in an appropriate way does solve the given problem efficiently, but overestimates the domination count probability and thus, does not yield good probability bounds. Rather, we have to redesign the generating functions technique such that lower/upper probability bounds can be handled correctly.

6.4.3 Uncertain Generating Functions (UGFs)

Given a set of N independent but not necessarily identically distributed Bernoulli $\{0, 1\}$ random variables $X_i, 1 \leq i \leq N$. Let $P_{LB}(X_i)$ ($P_{UB}(X_i)$) be a lower (upper) bound approximation of the probability $P(X_i = 1)$. Consider the random variable

$$\sum_{i=1}^{N} X_i.$$

We make the following observation: The lower and upper bound probabilities $P_{LB}(X_i)$ and $P_{UB}(X_i)$ correspond to the probabilities of the three following events:

- $X_i = 1$ definitely holds with a probability of at least $P_{LB}(A_i \prec_R B)$.
- $X_i = 0$ definitely holds with a probability of at least $1 - P_{UB}(X_i)$.
- It is unknown whether $X_i = 0$ or $X_i = 1$ with the remaining probability of $P_{UB}(A_i \prec_R B) - P_{LB}(A_i \prec_R B) = P_{UB}(A_i \prec_R B) - P_{LB}(A_i \prec_R B)$.

Based on this observation, we consider the following uncertain generating function (UGF):

$$\mathcal{F}^N = \prod_{i \in 1,\ldots,N} [(P_{LB}(X_i) \cdot x + (1 - P_{UB}(X_i)) \cdot y + (P_{UB}(X_i) - P_{LB}(X_i)))] = \sum_{i,j \geq 0} c_{i,j} x^i y^j.$$

The coefficient $c_{i,j}$ has the following meaning: With a probability of $c_{i,j}$, B is definitely dominated at least i times, and possibly dominated another 0 to j times. Therefore, the minimum probability that $\sum_{i=1}^{N} X_i = k$ is $c_{k,0}$, since that is the probability that exactly k random variables X_i are 1. The maximum probability that $\sum_{i=1}^{N} X_i = k$ is $\sum_{i \leq k, i+j \geq k} c_{i,j}$, i.e. the total probability of all possible combinations in which $\sum_{i=1}^{N} X_i = k$, may hold. Therefore, we obtain an approximated PDF of $\sum_{i=1}^{N} X_i$. In the approximated PDF of $\sum_{i=1}^{N} X_i$, each probability $\sum_{i=1}^{N} X_i = k$ is given by a conservative and a progressive approximation.

Example 18. *Let $P_{LB}(X_1) = 20\%$, $P_{UB}(X_1) = 50\%$, $P_{LB}(X_2) = 60\%$ and $P_{UB}(X_2) = 80\%$. The generating function for the random variable $\sum_{i=1}^{2} X_i$ is the following:*

$$\mathcal{F}^2 = (0.2x + 0.3y + 0.5)(0.6x + 0.2y + 0.2)$$
$$= 0.12x^2 + 0.34x + 0.1 + 0.22xy + 0.16y + 0.06y^2$$

That implies that, with a probability of at least 12%, $\sum_{i=1}^{2} X_i = 2$. In addition, with a probability of 22% plus 6%, it may hold that $\sum_{i=1}^{2} X_i = 2$, so that we obtain a probability bound of $12\% - 40\%$ for the random event $\sum_{i=1}^{2} X_i = 2$. Analogously, $\sum_{i=1}^{2} X_i = 1$ with a probability of $34\% - 78\%$ and $\sum_{i=1}^{2} X_i = 0$ with a probability of $10\% - 32\%$. The approximated PDF of $\sum_{i=1}^{2} X_i$ is depicted in Figure 6.4.

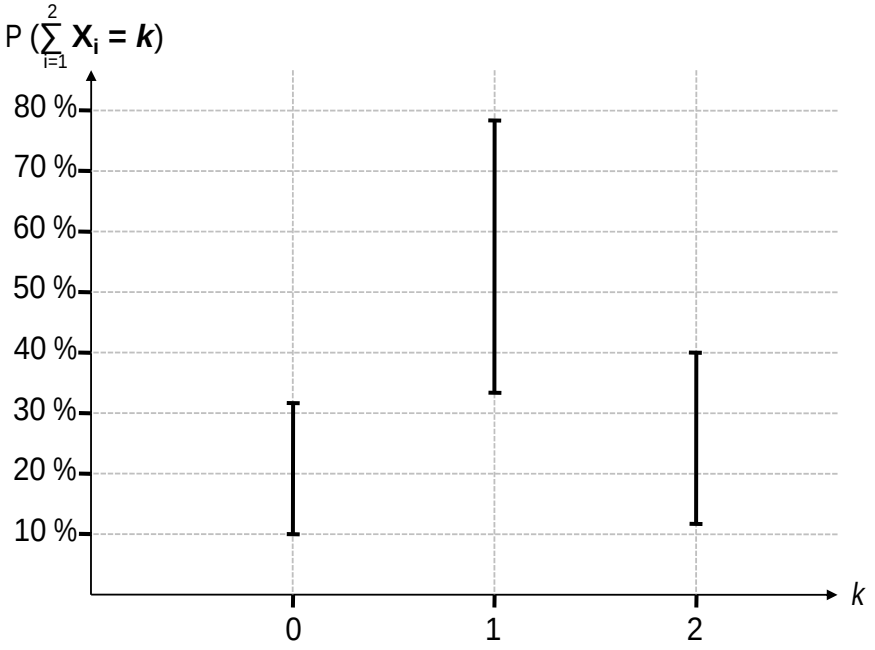


Figure 6.4: Approximated PDF of $\sum_{i=1}^{2} X_i$.

Each expansion $\mathcal{F}^l$ can be obtained from the expansion of $\mathcal{F}^{l-1}$ as follows:

$$\mathcal{F}^l = \mathcal{F}^{l-1}.$$
$$[P_{LB}(X_l) \cdot x + (1 - P_{UB}(X_l)) + (P_{UB}(X_l) - P_{LB}(X_l)) \cdot y].$$

We note that $\mathcal{F}^l$ contains at most $\sum_{i=1}^{l+1} i$ non-zero terms (one $c_{i,j}$ for each combination of i and j where $i + j \leq l$). Therefore, the total complexity to compute $\mathcal{F}^l$ is $O(l^3)$.

6.4.4 Efficient Domination Count Approximation using UGFs

We can directly use the uncertain generating functions proposed in the previous section to derive bounds for the probability distribution of the domination count $DomCount(B, Q)$. Again, let $\mathcal{DB} = A_1, ..., A_N$ be an uncertain object database and B and Q be uncertain objects in $\mathbb{R}^d$. Let $(A_i \prec_Q B), 1 \leq i \leq N$ denote the random Bernoulli event that A_i dominates B w.r.t. Q. Also recall that the domination count is defined as the random variable that is the sum of the domination indicator variables of all uncertain objects in the database (cf. Definition 32).

Considering the generating function

$$\mathcal{F}^N = \prod_{i \in 1,...,N} [(P_{LB}(A_i \prec_Q B) \cdot x +$$
$$(P_{UB}(A_i \prec_Q B) - P_{LB}(A_i \prec_Q B)) \cdot y) +$$

$$(1 - P_{UB}(A_i \prec_Q B))] = \sum_{i,j \geq 0} c_{i,j} x^i y^j, \tag{6.3}$$

we can efficiently compute lower and upper bounds of the probability that $DomCount(B, Q) = k$ for $0 \leq k \leq |\mathcal{DB}|$, as discussed in Section 6.4.3 and because the independence property of random variables required by the generating functions is satisfied due to Lemma 19.

Lemma 20. *A lower bound $DomCount^k_{LB}(B, Q)$ of the probability that $DomCount(B, Q) = k$ is given by*

$$DomCount^k_{LB}(B, Q) = c_{k,0}$$

and an upper bound $DomCount^k_{UB}(B, Q)$ of the probability that $DomCount(B, Q) = k$ is given by

$$DomCount^k_{UB}(B, Q) = \sum_{i \leq k, i+j \geq k} c_{i,j}$$

Example 19. *Assume a database containing uncertain objects A_1, A_2, B and Q. The task is to determine a lower (upper) bound of the domination count probability $DomCount^k_{LB}(B, Q)$ ($DomCount^k_{UB}(B, Q)$) of B w.r.t. Q. Assume that, by decomposing A_1 and A_2 and using the probabilistic domination approximation technique proposed in Section 6.3.2, we determine that A_1 has a minimum probability $P_{LB}(A_1 \prec_Q B)$ of dominating B of 20% and a maximum probability $P_{UB}(A_1 \prec_Q B)$ of 50%. For A_2, $P_{LB}(A_2 \prec_Q B)$ is 60% and $P_{UB}(A_2 \prec_Q B)$ is 80%. By applying the technique in the previous subsection, we get the same generating function as in Example 18 and thus, the same approximated PDF for the $DomCount(B, Q)$ depicted in Figure 6.4.*

To compute the uncertain generating function and thus the probabilistic domination count of an object in an uncertain database of size N, the total complexity is $O(N^3)$. The reason is that the maximal number of coefficients of the generating function $\mathcal{F}^x$ is quadratic in x, since $\mathcal{F}^x$ contains coefficients $c_{i,j}$ where $i+j \leq x$, that is at most $\frac{x^2}{2}$ coefficients. Since we have to compute $\mathcal{F}^x$ for each ($x < N$), the total time complexity is $O(N^3)$. Note that only candidate objects $c \in Cand$ for which a complete domination cannot be detected (cf. Section 6.3.1) have to be considered in the generating functions. Thus, the total runtime to compute $DomCount^k_{LB}(B, Q)$ as well as $DomCount^k_{UB}(B, Q)$ is $O(|Cand|^3)$. Finally, we will show how to reduce, specifically for kNN queries, the total time complexity to $O(k^2 \cdot |Cand|)$.

6.4.5 Generating Functions vs Uncertain Generating Functions

It is clear that instead of applying the uncertain generating function to approximate the domination count of B, two regular generating functions can be used; one generating function that uses the progressive (lower) bounds $P_{UB}(A_i \prec_Q B)$ and one that uses the conservative (upper) probability bounds $P_{UB}(A_i \prec_Q B)$. In the following we give an intuition and a formal proof that using regular generating functions yields looser bounds for the approximated domination.

Let $\mathcal{DB} = A_1, ..., A_N$ be an uncertain object database and B and Q be uncertain objects in $\mathbb{R}^d$. Let $(A_i \prec_Q B), 1 \leq i \leq N$ denote the random Bernoulli event that A_i dominates B w.r.t. Q. Let $P_{LB}(A_i \prec_Q B)$ ($P_{UB}(A_i \prec_Q B)$) be a lower (upper) bound approximation of the probabilistic event $(A_i \prec_Q B) = 1$.

A lower bound of the probability $DomCount(B, Q) = k$ can be derived using the following generating function:

$$\mathcal{F}^N = \prod_{i \in 1,...,N} [(P_{LB}(A_i \prec_Q B) \cdot x + (1 - P_{UB}(A_i \prec_Q B))] = \sum_{i \geq 0} c_i x^i. \tag{6.4}$$

Intuitively, this generating function uses the progressive approximation $P_{LB}(A_i \prec_Q B)$ of the probability that A_i dominates B w.r.t. Q and the progressive approximation $1 - P_{UB}(A_i \prec_Q B)$ of the probability that A_i does not dominate B w.r.t. Q. This generating function is equal to the uncertain generating function (cf. Equation 6.3) if the uncertain percentage (i.e. the coefficient of y) is omitted for each candidate A_i, i.e. if the coefficient of each y is set to 0.

Lemma 21. *Let c_k be the coefficients obtained by the generating function in Equation 6.4 and let $P_{LB}(DomCount(B, Q) = k)$ be the lower bound derived by applying the generating function in Equation 6.3 and exploiting Lemma 20. It holds that*

$$c_k = P_{LB}(DomCount(B, Q) = k),$$

i.e. the lower bound obtained by the (non-uncertain) generating function in Equation 6.4 is as good as the lower bound obtained using the technique in Section 6.4.4.

Proof. The lower bound derived using the uncertain generating function for the probability $DomCount(B, Q) = k$ is equal to the coefficient $c_{k,0}$. The coefficient $c_{k,0}$ corresponds to the variable $x^k y^0 = x^k$. Since Equation 6.4 and Equation 6.3 are identical except for the variables containing at least one y, but no y is contained in the variable of the coefficient $c_{k,0}$ in Equation 6.3, it is identical to the coefficient c_k in Equation 6.4. □

We can see that we can use a (non-uncertain) generating function to derive the same lower bound. The advantage here is that the (non-uncertain) generating function is easier to compute, due to a much (linear in k) lower number of coefficients. However, deriving an upper bound of the probability $DomCount(B, Q) = k$ is not as easy, because the upper bound does use the uncertainty of objects. An upper bound can be derived using the following generating function:

$$\mathcal{F}^N = \prod_{i \in 1,...,N} [(P_{UB}(A_i \prec_Q B) \cdot x + (1 - P_{LB}(A_i \prec_Q B))] = \sum_{i \geq 0} c_i x^i. \tag{6.5}$$

The idea is to use a conservative approximation $P_{UB}(A_i \prec_Q B)$ for the probability that A_i dominates B and a conservative approximation $1 - P_{LB}(A_i \prec_Q B)$ for the probability that A_i does not dominate B. The difference of this generating function compared to the uncertain generating function (cf. Equation 6.3) is that the uncertain percentage (the coefficient of y) is added to both probabilities $P(A_i \prec_Q B)$ and $P(\neg(A_i \prec_Q B))$[3].

Lemma 22. *Let c_k be the coefficients obtained by the generating function in Equation 6.5 and let $P_{LB}(DomCount(B, Q) = k)$ be the lower bound derived by applying the generating function in Equation 6.3 and exploiting Lemma 20. It holds that*

$$c_k \geq P_{LB}(DomCount(B, Q) = k),$$

i.e. the upper bound obtained by the (non-uncertain) generating function in Equation 6.4 is in general not as good as the lower bound obtained using the technique in Section 6.4.4.

Proof. We give an example where the upper bound derived by Equation 6.4 is worse than the upper bound derived by Equation 6.3 and exploiting Lemma 20. Assume a database containing uncertain objects A_1, A_2, B and Q. The task is to determine the probability that $DomCount(B, Q) = 1$. Also assume that we have determined (e.g. as proposed in Section 6.3.2) a lower bound $P_{LB}(A_1 \prec_Q B)$ ($P_{LB}(A_2 \prec_Q B)$) and an upper bound $P_{UB}(A_1 \prec_Q B)$ ($P_{UB}(A_2 \prec_Q B)$) of the probability that A_1 (A_2) dominates B w.r.t. Q. To ease the notation, let A_i^+ denote $P_{LB}(A_i \prec_Q B)$, let A_i^- denote $1 - P_{UB}(A_i \prec_Q B)$ and let $A_i^?$ denote the uncertain fraction $P_{UB}(A_i \prec_Q B) - P_{LB}(A_i \prec_Q B)$.

[3]Note that adding the coefficient of y to only one of the summands will result in incorrect bounds.

Using this notation, Equation 6.3 becomes:

$$(A^+x + A^?y + A^-) \cdot (B^+x + B^?y + B^-)$$

Expansion yields:

$$A^+B^+x^2 + A^+B^?xy + A^+B^-x + A^?B^+xy + A^?B^?y^2 +$$

$$A^?B^-y + A^-B^+x + A^-B^?y + A^-B^-$$

Exploiting Lemma 20 yields the following upper bound probability for $DomCount(B, Q) = 1$:

$$P_{UB}(DomCount(B, Q) = 1) =$$
$$A^+B^? + A^+B^- + A^?B^+ + A^?B^? + A^-B^? + A^-B^+ + A^-B^?$$

On the other hand, Equation 6.4 becomes:

$$((A^+ + A^?)x + A^- + A^?) \cdot ((B^+ + B^?)x + B^- + B^?)$$

Expansion yields:

$$(A^+ + A^?) \cdot (B^+ + B^?)x^2 + (A^+ + A^?) \cdot (B^- + B^?)x +$$

$$(A^- + A^?) \cdot (B^+ + B^?)x + (A^- + A^?) \cdot (B^- + B^?)$$

Extracting the coefficient c_1 of x^1 yields:

$$c_1 = (A^+ + A^?) \cdot (B^- + B^?) + (A^- + A^?) \cdot (B^+ + B^?)$$

Expansion yields:

$$c_1 = A^+B^- + A^+ + B^? + A^?B^- + A^?B^? +$$
$$A^-B^+ + A^-B^? + A^?B^+ + A^?B^?$$

Comparing $P_{UB}(DomCount(B, Q) = 1)$ and c_1, we obtain:

$$c_1 - P_{UB}(DomCount(B, Q) = 1) = A^?B^?$$

Thus, the upper bound c_1 of the (non-uncertain) generating function is greater (and thus worse) than the upper bound $P_{UB}(DomCount(B, Q) = 1)$ of the uncertain generating function by a total of $A^?B^?$. □

Intuitively, the problem of the upper bound using the (non-uncertain) generating function is that the uncertain fraction (i.e. $A^?$) is added to both the probability that A_i dominates B (i.e. A^+) and to the probability that A_i does not dominate B (i.e. A^-). Therefore, this approach incorrectly considers some possible worlds more often than once.

6.4.6 Efficient Domination Count Approximation Based on Disjunctive Worlds

Since the uncertain objects B and Q appear in each domination relation $P(A_1 \prec_Q B)$, ..., $P(A_C \prec_Q B)$ that is to evaluate, we cannot split objects B and Q independently (cf. Section 6.4.1). The reason for this dependency is that knowledge about the predicate $(A_i \prec_Q B)$ may impose constraints on the position of B and Q. Thus, for a partition $B_1 \subset B$, the probability $P(A_j \prec_Q B_1)$ may change given $(A_i \prec_Q B)$ $(1 \leq i, j \leq C, i \neq j)$. However, note:

Lemma 23. *Given fixed partitions $B' \subseteq B$ and $Q' \subseteq Q$, then the random variables $(A_i \prec_{Q'} B')$ are mutually independent for $1 \leq i, j \leq C, i \neq j$.*

Proof. The proof of this lemma is analogous to the proof of Lemma 19. □

This allows us to individually consider the subset of possible worlds where $b \in B'$ and $r \in Q'$ and use Lemma 23 to efficiently compute the approximated domination count probabilities $DomCount^k_{LB}(B', Q')$ and $DomCount^k_{UB}(B', Q')$ under the condition that B falls into a partition $B' \subseteq B$ and Q falls into a partition $Q' \subseteq Q$. This can be performed for each pair $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$, where $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ denote the decompositions of B and Q, respectively. Now, we can treat pairs of partitions $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$ independently, since all pairs of partition represent disjunctive sets of possible worlds due to the assumption of a disjunctive partitioning. Exploiting this independency, the PDF of the domination count $DomCount(B, Q)$ of the total objects B and Q can then be obtained by creating an uncertain generating function for each pair (B', Q') to derive a lower and an upper bound of $P(DomCount(B', Q') = k)$ and then computing the weighted sum of these bounds as follows:

$$DomCount^k_{LB}(B, Q) = \sum_{B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} DomCount^k_{LB}(B', Q') \cdot P(B') \cdot P(Q').$$

The complete algorithm of our domination count approximation approach can be found in the next Section.

6.5 Implementation

Algorithm 1 Probabilistic Domination Count Algorithm

Require: : Q, B, $\mathcal{DB}$
1: $influenceObjects = \emptyset$; $CompleteDominationCount = 0$
2: *//Complete Domination*
3: **for all** $A_i \in \mathcal{DB}$ **do**
4: **if** $DDC_{Optimal}(A_i, B, Q)$ **then**
5: $CompleteDominationCount$++
6: **else if** $\neg DDC_{Optimal}(B, A_i, Q)$ **then**
7: $influenceObjects = influenceObjects \cap A_i$
8: **end if**
9: **end for**
10: *//probabilistic domination count*
11: $DomCount_{LB}$= [0,...,0] //length $|\mathcal{DB}|$
12: $DomCount_{UB}$= [1,...,1] //length $|\mathcal{DB}|$
13: **while** ¬ stopcriterion **do**
14: split(Q), split(B), split($A_i \in \mathcal{DB}$)
15: **for all** $B' \in B$, $Q' \in Q$ **do**
16: $cand_{LB}$= [0,...,0] //length $|uncertainObjects|$
17: $cand_{UB}$= [1,...,1] //length $|uncertainObjects|$
18: **for all** $(0 < i < |influenceObjects|)$ **do**
19: $A_i = influenceObjects[i]$
20: **for all** $A'_i \in A_i$ **do**
21: **if** $DDC_{Optimal}(A'_i, B', Q')$ **then**
22: $cand_{LB}[i]$+=$(P(A'_i))$
23: **else if** $DDC_{Optimal}(B', A'_i, Q')$ **then**
24: $cand_{UB}[i]$-=$(P(A'_i))$
25: **end if**
26: **end for**
27: **end for**
28: compute $DomCount_{LB}(B', Q')$ and $DomCount_{UB}(B', Q')$ using UGFs.
29: **for all** $(0 < i < \mathcal{DB})$ **do**
30: $DomCount_{LB}[i]$+=$DomCount(B', Q')_{LB} \cdot P(B') \cdot P(Q')$
31: $DomCount_{UB}[i]$+=$DomCount(B', Q')_{UB} \cdot P(B') \cdot P(Q')$
32: **end for**
33: **end for**
34: ShiftRight($DomCount_{LB}$,$CompleteDominationCount$)
35: ShiftRight($DomCount_{UB}$,$CompleteDominationCount$)
36: **end while**
37: return ($DomCount_{LB}$, $DomCount_{UB}$)

Algorithm 1 is a complete method for iteratively computing and refining the probabilistic domination count for a given object B and a query object Q. The algorithm starts by detecting complete domination (cf. Section 6.3.1). For each object that completely dominates B, a counter $CompleteDominationCount$ is increased and each object that is completely dominated by B is removed from further consideration, since it has no influence on the domination count of B. The remaining objects, which may have a probability greater than zero and less than one to dominate B, are stored in a set $influenceObjects$. The set $influenceObjects$ is now used to compute the probabilistic domination count ($DomCount_{LB}$, $DomCount_{UB}$)[4]: The main loop of the probabilistic domination count approximation starts in line 13. In each iteration, B, Q, and all influence objects are partitioned. For each combination of partitions B' and Q', and each database object $A_i \in influenceObjects$, the probability $P(A_i \prec_{Q'} B')$ is approximated (cf. Section 6.4.2). These domination probability bounds are used to build an uncertain generating function (cf. Section 6.4.4) for the domination count of B' w.r.t. Q'. Finally, these domination counts are aggregated for each pair of partitions B', Q' into the domination count $DomCount(B, Q)$ (cf. Section 8.5.3). The main loop continues until a domain- and user-specific stop criterion is satisfied. For example, for a threshold kNN query, a stop criterion is to decide whether the lower (upper) bound that B has a domination count of less than (at least) k, exceeds (falls below) the given threshold.

The progressive decomposition of objects (line 14) can be facilitated by precomputed split points at the object PDFs. More specifically, we can iteratively split each object X by means of a median-split-based bisection method and use a kd-tree [17] to hierarchically organize the resulting partitions. The kd-tree is a binary tree. The root of a kd-tree represents the complete region of an uncertain object. Every node implicitly generates a splitting hyperplane that divides the space into two subspaces. This hyperplane is perpendicular to a chosen split axis and located at the median of the node's distribution in this axis. The advantage is that, for each node in the kd-tree, the probability of the respective subregion X' is simply given by $0.5^{X'.level-1}$, where $X'.level$ is the level of X'. In addition, the bounds of a subregion X' can be determined by backtracking to the root. In general, for continuously partitioned uncertain objects, the corresponding kd-tree may have an infinite height, however for practical reasons, the height h of the kd-tree is limited. The choice of h is a trade-off between approximation quality and efficiency: for a very large h, considering each leaf node is similar to applying integration on the PDFs, which yields an exact result; however, the number of leaf nodes, and thus the worst case complexity increases exponentially in h. Note that our experiments (c.f. Section 6.6) show that a low h value is sufficient to yield reasonably tight approximation bounds. Yet it has to be noted, that in the general case of continuous uncertainty, our proposed approach may only return an approximation of the exact probabilistic domination count. However, such an approximation may be sufficient to decide a given predicate as we will also see in Section

[4] $DomCount_{LB}$ and $DomCount_{UB}$ are lists containing, at each position i, a lower and an upper bound for $P(DomCount(B, Q) = i)$, respectively. This notation is equivalent to a single uncertain domination count PDF.

6.6 and even in the case where the approximation does not suffice to decide the query predicate, the approximation will give the user a confidence value, based on which a user may be able decide whether to include an object in the result.

For kNN queries, the total complexity to compute the uncertain generating function can be improved from $O(|Cand|^3)$ to $O(|Cand| \cdot k^2)$ since it can be observed from Corollary 5 that for kNN queries, we only require the section of the PDF of $DomCount(B, Q)$ where $DomCount(B, Q) < k$, i.e. we only need to know the probabilities $P(DomCount(B, Q) = x), x < k$. This can be exploited to improve the runtime of the computation of the PDF of $DomCount(B, Q)$ as follows: Consider the iterative computation of the generating functions $\mathcal{F}^1, ..., \mathcal{F}^{|cand|}$. For each $\mathcal{F}^l, 1 \leq l \leq |cand|$, we only need to consider the coefficients $c_{i,j}$ in the generating function $\mathcal{F}^i$ where $i < k$, since only these coefficients have an influence on $P(DomCount(B, Q) = x), x < k$ (cf. Section 20). In addition, we can merge all coefficients $c_{i,j}$, $c_{i',j'}$ where $i = i'$, $i + j > k$ and $i' + j' > k$, since all these coefficients only differ in their influence on the upper bounds of $P(DomCount(B, Q) = x), x \geq k$, and are treated equally for $P(DomCount(B, Q) = x), x < k$. Thus, each $\mathcal{F}^l$ contains at most $\sum_{i=1}^{k+1} i$ coefficients (one $c_{i,j}$ for each combination of i and j where $i + j \leq k$). Thus reducing the total complexity to $O(k^2 \cdot |cand|)$.

6.6 Experimental Evaluation

In this section, we review the characteristics of the proposed algorithm on synthetic and real-world data. The algorithm will be referred to as **IDCA** (Iterative Domination Count Approximation). We performed experiments under various parameter settings. Unless otherwise stated, for 100 queries, we chose B to be the object with the 10^{th} smallest MinDist to the query object Q. We used a synthetic dataset with 10,000 objects modeled as 2D rectangles. The degree of uncertainty of the objects in each dimension is modeled by their relative extent. The extents were generated uniformly and at random with 0.004 as maximum value. For the evaluation on real-world data, we utilized the International Ice Patrol (IIP) Iceberg Sightings Dataset[5]. This dataset contains information about iceberg activity in the North Atlantic in 2009. The latitude and longitude values of sighted icebergs serve as certain 2D mean values for the 6,216 probabilistic objects that we generated. Based on the date and the time of the latest sighting, we added Gaussian noise to each object, such that the passed time period since the latest date of sighting corresponds to the degree of uncertainty (i.e. the extent). The extents were normalized w.r.t. the extent of the data space, and the maximum extent of an object in either dimension is 0.0004.

6.6.1 Runtime of the Monte-Carlo-based Approach

To the best of our knowledge, there exists no approach which is able to process uncertain similarity queries on probabilistic databases with continuous PDFs. A naive approach

[5]The IIP dataset is available at the National Snow and Ice Data Center (NSIDC) web site (*http://nsidc.org/data/g00807.html*).

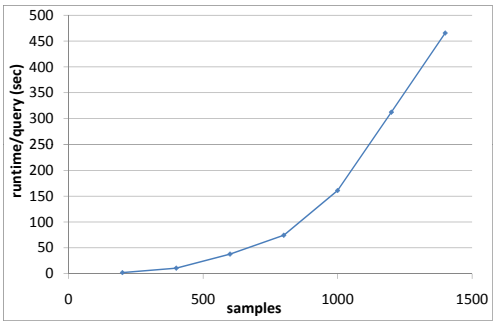
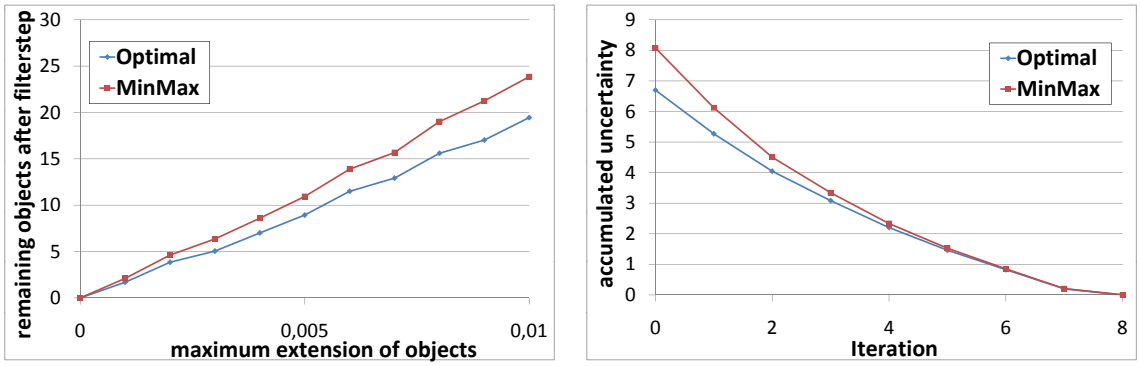


Figure 6.5: Runtime of **MC** for increasing sample size.



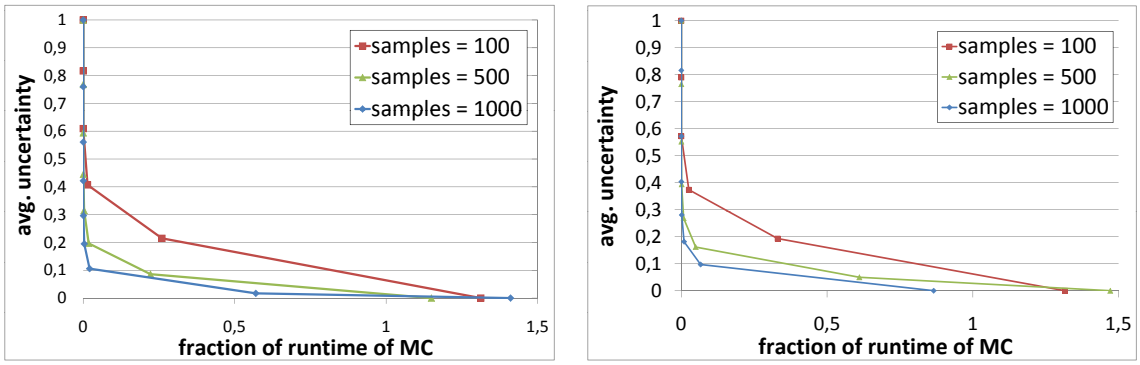
(a) Candidates after spatial pruning.

(b) Accumulated uncertainty of result.

Figure 6.6: Optimal vs. MinMax decision criterion.

needs to consider all possible worlds and thus needs to integrate over all object PDFs, implying a runtime exponentially in the number of objects. Since this is not applicable even for small databases, we adapted an existing approach to cope with the conditions. The approach most related to our work is [131], which solves the problem of computing the domination count for a certain query and discrete distributions within the database objects. Thus the proposed comparison partner works as follows: Draw a sufficiently large number S of samples from each object by Monte-Carlo-Sampling. Then, for each sample $q_i \in Q$ of the query, apply the algorithm proposed in [131] to compute an exact probabilistic domination count PDF of an object B. As proposed in [131], this is done using the generating function technique and using an *and/xor tree* to combine individual samples into discrete distributed uncertain objects. Finally, accumulate the resulting certain domination count PDFs of each $q_i \in Q$ into a single domination count PDF by taking the average. The execution time for this approach, which we will refer to as **MC** in the following, is shown in Figure 6.5. It can be observed that for a reasonable sample size (which is required to achieve a result that is close to the correct result with high probability) the runtime becomes very large.

Note that our comparison partner only works for discrete uncertain data (cf. Section 6.6.1). To make a fair comparison our approach relies on the same uncertainty model (default: 1000 samples/object). Nevertheless, all the experiments yield analogous results for continuous distributions.



(a) Synthetic Data (b) Real Data

Figure 6.7: Uncertainty of **IDCA** w.r.t. the relative runtime to **MC**.

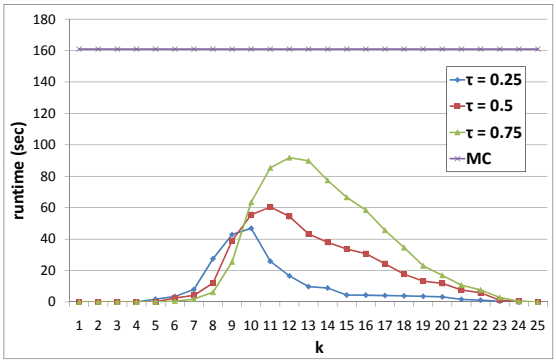
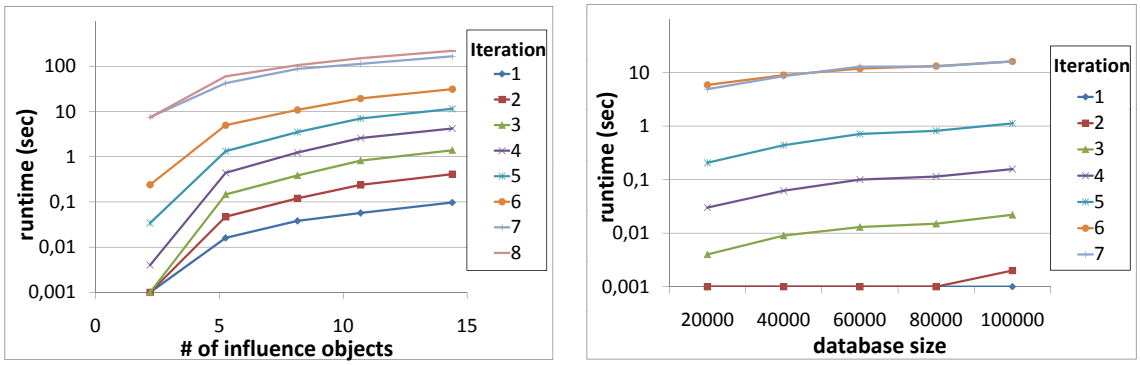


Figure 6.8: Runtimes of **IDCA** and **MC** for different query predicates k and τ.

6.6.2 Optimal vs. Min/Max Decision Criterion

In the first experiment, we evaluate the gain of pruning power using the complete similarity domination technique (cf. Section 6.3.1) instead of the state-of-the-art min/max decision criterion to prune uncertain objects from the search space. The first experiment evaluates the number of uncertain objects that cannot be pruned using complete domination only, that is the number of candidates are to evaluate in our algorithm. Figure 6.6(a) shows that our domination criterion (in the following denoted as optimal) is able to prune about 20% more candidates than the min/max pruning criterion. In addition, we evaluated the domination count approximation quality (in the remainder denoted as uncertainty) after each decomposition iteration of the algorithm, which is defined as the sum $\sum_{i=0}^{N} DomCount^i_{UB}(B,Q) - DomCount^i_{LB}(B,Q)$. The result is shown in Figure 6.6(b). The improvement of the complete domination (denoted as iteration 0) can also be observed in further iterations. After enough iterations, the uncertainty converges to zero for both approaches.



(a) Runtime w.r.t. number of influence objects. (b) Runtime for different sizes of the database.

Figure 6.9: Impact of influencing objects.

6.6.3 Iterative Domination Count Approximation

Next, we evaluate the trade-off of our approach regarding approximation quality and the invested runtime of our domination count approximation. The results can be seen in Figure 6.7 for different sample sizes and datasets. It can be seen that initially, i.e. in the first iterations, the average approximation quality (avg. uncertainty of an *influenceObject*) decreases rapidly. The less uncertainty left, the more computational power is required to reduce it any further. Except for the last iteration (resulting in 0 uncertainty) each of the previous iterations is considerably faster than **MC**. In some cases (see Figure 6.7(b)) **IDCA** is even faster in computing the exact result.

6.6.4 Queries with a Predicate

Integrated in an application one often wants to decide whether an object satisfies a predicate with a certain probability. In the next experiment, we posed queries in the form: Is object B among the k nearest neighbors of Q (predicate) with a probability of 25%, 50%, 75%? The results are shown in Figure 6.8 for various k-values. With a given predicate, **IDCA** is often able to terminate the iterative refinement of the objects earlier in most of the cases, which results in a runtime which is orders of magnitude below **MC**. In average the runtime is below **MC** in all settings.

6.6.5 Number of influenceObjects

The runtime of the algorithm is mainly dependent on the number of objects which are responsible for the uncertainty of the rank of B. The number of *influenceObjects* depends on the number of objects in the database, the extension of the objects and the distance between Q and B. The larger this distance, the higher the number of *influenceObjects*. For the experiments in Figure 6.9(a) we varied the distance between Q and B and measured the runtime for each iteration. In Figure 6.9(b) we present runtimes for different sizes of

the database. The maximum extent of the objects was set to 0.002 and the number of objects in the database was scaled from 20,000 to 100,000. Both experiments show that **IDCA** scales well with the number influencing objects.

6.7 Conclusions

In this chapter, we applied the concept of probabilistic similarity domination on uncertain data. We introduced a geometric pruning filter to conservatively and progressively approximate the probability that an object is being dominated by another object. An iterative filter-refinement strategy is used to stepwise improve this approximation in an efficient way. Specifically we propose a method to efficiently and effectively approximate the domination count of an object using a novel technique of uncertain generating functions. We show that the proposed concepts can be used to efficiently answer probabilistic k-nearest neighbor queries while keeping correctness according to the possible world semantics. Our experiments show that our iterative filter-refinement strategy is able to achieve a high level of precision at a low runtime.

Chapter 7

Probabilistic Ranking on Uncertain Data

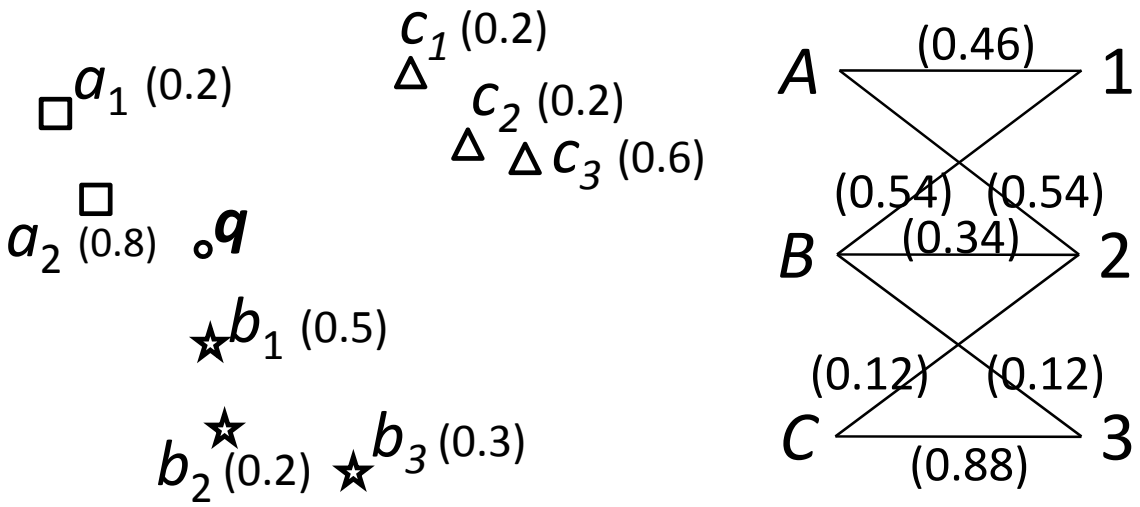
This chapter introduces a scalable approach for probabilistic similarity ranking on uncertain vector data. Uncertain objects are modelled using the discrete X-tuple model. The objective is to rank the uncertain data according to their distance to a reference object. We propose a framework that incrementally computes for each object instance and ranking position, the probability of the object falling at that ranking position. The resulting rank probability distribution can serve as input for several state-of-the-art probabilistic ranking models. Existing approaches compute this probability distribution by applying the *Poisson binomial recurrence* technique of quadratic complexity. In this chapter we theoretically as well as experimentally show that our framework reduces this to a linear-time complexity while having the same memory requirements, facilitated by incremental accessing of the uncertain vector instances in increasing order of their distance to the reference object. Furthermore, we show how the output of our method can be used to apply probabilistic ranking for the objects, according to different state-of-the-art definitions. We conduct an experimental evaluation on synthetic and real data, which demonstrates the efficiency of our approach.

7.1 Introduction

Similarity ranking is a hot topic in database research because it plays a major role in a large number of emerging applications, such as data retrieval, decision support systems, and data mining that require exploratory querying of uncertain databases. For example, clustering and ranking have a mutual reinforcement property for search engines. While search engines use clustering to identify groups of relevant objects, ranking is used to report the most important first. A ranking query orders the objects in a database with respect to their similarity to a reference object. In a spatial database context, ranking queries return the contents of a spatial object set (e.g., restaurants) in increasing order of their distance to a reference location. In a database of images, a similarity query ranks the feature vectors of images in increasing order of their distance (i.e., dissimilarity) to a query image.

In this chapter, we focus on similarity ranking of uncertain vector data. Prior work in this direction includes [36, 55, 172, 171, 203]. For the special case where all objects are represented by a Gaussian distribution, a solution for probabilistic ranking is given by [36]. However, this solution cannot be extended to handle arbitrary distributions, as this approach involves expensive integrations of the pdf's. All other mentioned publication, employ the x-relations model used in the *Trio* system [7]. In this chapter, we once more adopt the same model.

Consider, for example, a set of three two-dimensional objects A, B, and C (e.g., locations of mobile users), and their corresponding uncertain instances $\{a_1, a_2\}$, $\{b_1, b_2, b_3\}$, and $\{c_1, c_2, c_3\}$, as shown in Figure 7.1(a). Each instance carries a probability (shown in brackets) and instances of the same object are mutually-exclusive. Assume that we wish to rank the objects A, B, and C according to their distances to the query point q shown in the figure. Clearly, several rankings are possible. In specific, each combination of object instances defines an order. For example, for combination $\{a_1, b_1, c_1\}$ the object ranking is (B, A, C) while for combination $\{a_2, b_3, c_1\}$ the object ranking is (A, B, C).



(a) Object Instances (b) Bipartite Graph

Figure 7.1: Object Instances and Rank Probability Graph

This example illustrates the ambiguity of ranking in uncertain data. However, most applications require the definition of a non-ambiguous object ranking. For example, assume that a robbery took place at location q and the objects correspond to the positions of suspects that are sampled around the time that the robbery took place. The probabilities of the samples depend on various factors (e.g., time-difference of the sample to the robbery event, errors of capturing devices, etc.). As an application, we may want to define a definite probabilistic proximity ordering of the suspects to the event, in order to prioritize interrogations.

Various top-k query approaches have been proposed generating un-ambiguous rankings from probabilistic data. Examples are U-top$k$ and U-kRanks [172], PT-k [91], Global top-k [213], and expected rank [55]. A summary of these ranking models can be found in Section 7.5. All of them attempt to weigh the objects based on their probability to be in each of the first k ranks, but they use different ways to define the weights.

A common module in most of these approaches, with the only exception being the expected ranks approach of [55], is the computation for each object instance x the probability P_i that i objects are closer to q than x for all $1 \leq i \leq k$. The resulting probabilities are aggregated to build the probability of each object at each rank. For example, the U-kRanks query reports the i^{th} result as the object that is the most likely to be ranked i^{th} over all possible worlds. For this computation, we obviously need the probabilities of all instances to be ranked i^{th} over all possible worlds. The probability that an object is ranked at a specific position i can be computed by summing the probabilities of the possible worlds that support this occurrence. In our example, the probability that object A occurs as first one is 0.46 and the probability that object B is the first is 0.54. All possible occurrences and the corresponding probabilities are represented by the object-rank bipartite graph which is shown in Figure 7.1(b). Non-existing edges imply zero probability, i.e. it is not possible that the object occurs at the corresponding ranking position. In this example, all instances of A precede all those of C, so C cannot occur as first object and A cannot be ranked to the last position.

In this chapter, we propose a framework that, given a database with uncertain vector objects, computes the rank probabilities of the object instances (e.g., a_1) in linear time to the total number of instances of all objects. Here we assume that the instances are accessed in increasing distance order to the query object q (e.g., with the help of a nearest neighbor search algorithm [88]). As these can be aggregated on-the-fly, our framework also computes the rank probabilities of the objects at the same cost. This is a great improvement, over the state-of-the-art [203], which computes these probabilities in quadratic time. Since the number of possible worlds is exponential in the number of uncertain objects, it is impractical to enumerate all of them in order to find the rank probabilities of all object instances. Recently, it has been shown in [203] that we can compute the probabilities between all object instances and ranks in $O(kn^2)$ time, where k is the maximum ranking position and n is the number of object instances required to be accessed until the solution is confirmed. This solution can be applied to all problems that comply to the x-relation model (including our problem). In this chapter, we propose a significant improvement of this approach, which reduces the time complexity to $O(kn)$.

7.1.1 Contributions and Outline

The main contributions of this chapter can be summarized as follows:

- **Framework:** We propose a framework based on iterative distance browsing that efficiently supports probabilistic similarity ranking in uncertain vector databases.

- **Probabilistic Ranking Module:** We present a novel and theoretically founded approach for computing the rank probabilities of each object. We prove that our method reduces the computational cost of the rank probabilities from $O(kn^2)$, achieved by the best currently known method, to $O(kn)$.
- **Ranking Queries on Probabilistic Data:** We show how diverse state-of-the-art probabilistic ranking models can use our framework to accelerate computation.
- **Experiments:** We conduct an experimental evaluation, using real and synthetic data, which demonstrates the applicability of our framework and verifies our theoretical findings.

The rest of the chapter is organized as follows: In the next section, we survey existing work in the field of managing and querying uncertain data. In Section 7.3, we introduce our framework for computing the rank probabilities of uncertain object instances, followed by the details regarding the efficient incremental rank probability computation for each object instance.

The complete algorithm for computing the rank probabilities for all instances and the corresponding objects is presented in Section 7.4. In Section 7.5, we discuss in detail how our method can be used as a module in various models that rank the objects according to the rank probabilities of their instances. We experimentally evaluate the efficiency of our approach in Section 7.6 and conclude the chapter in Section 7.7.

7.2 Related Work

An initial approach for probabilistic ranking in uncertain databases was presented by [129]. This work proposes spatial and probabilistic pruning strategies, which allow to identify object that cannot possibly have a sufficiently high probability to be the top-k closest object to a query point. However, this work does not present an efficient solution for the refinement of the remaining candidate objects that have not been pruned. Instead, a straight-forward approach is presented which enumerates all possible worlds to compute exact probabilities for an object to be at a ranking position. One of the first approaches to improve this computation for probabilistic ranking queries is [27]. In this work, a divide and conquer method for accelerating the computation of the ranking probabilities is proposed. Although the proposed approach achieves a significant speed-up compared to the naive solution incorporating each possible database instance, its worst-case runtime is still exponential. Related to our ranking problem, significant work has been done in the field of probabilistic top-k query processing. Soliman et al. [172] were the first who studied such problems on the x-relations model of [16]. They proposed two ways of ranking uncertain tuples. In the first, *uncertain top-k* (U-Top$k$) query, the objective is to find the k-permutation of the most likely tuples to be the top-k. In our setting, this corresponds to finding the top-k most probable object instances (belonging to different objects) in all possible worlds. The *uncertain k-ranks query* (U-kRanks) reports a probabilistic ranking of

the tuples (again, *not* the x-tuples). The proposed solution shows a run-time exponential in the number of ranked objects k. This inefficient run-time can be explained by the probabilistic result semantics (see Section 2.4): Unlike the works of [27, 55, 171, 203], and unlike this chapter, the work of [172] uses result based query semantics (c.f. Section 2.4.2). For this reason, the number of possible results having a non-zero probability is in $O(|\mathcal{DB}|^k)$. Nevertheless, [172] is able to give an efficient solution for the special case where all tuples are mutually independent. This assumption however does not generally hold in the x-relation model, where tuples of the same x-relation are mutual exclusive.

At the same time Re et al. proposed in [155] an efficient but approximative probabilistic ranking based on Monte-Carlo sampling. Later, Yi et al. proposed in [203] the first efficient exact probabilistic ranking approach for the x-relation model, for both cases of single-alternative x-tuples only, i.e. x-tuples with only one uncertain instance, and multi-alternative x-tuples. They proposed Poisson-binomial-recurrence based methods for the computation of uncertain ranking queries, which have much lower costs than the previously best known results. Furthermore, they proposed early stopping conditions for accessing the tuples. Their methods for U-Top$k$ and U-kRanks queries have O($nlogk$) and O(kn^2) time complexity, respectively. The cost of the U-kRanks algorithm is dominated by the computation of the probability of each accessed tuple to be in each of the k first ranks. In this chapter, we also use this as a module of finding the object-rank probabilities. However, we propose an improvement of their O(kn^2) algorithm that does the same work in O(kn) without increasing the memory requirements.

In a recent paper, Cormode et al. [55] reviewed alternative top-k ranking approaches for uncertain data, including the U-Top$k$ and U-kRanks queries, and argued for a more robust definition of ranking, namely the *expected* rank for each tuple (or x-tuple). This is defined by the weighted sum of the ranks of the tuple in all possible worlds, where each world in the sum is weighed by its probability. The k tuples with the lowest expected ranks are argued to be a more appropriate definition of a top-k query than previous approaches. Nevertheless, we found by experimentation that such a definition may not be appropriate for ranking objects (i.e., x-tuples), whose instances have large variance (i.e., they are scattered far from each other in space). In general, the result of this ranking method is similar to the brute-force approach that would take the mean of the instances for each object and rank these means. On the other hand, approaches that take into consideration the rank probabilities (e.g., U-kRanks) would be more suitable for such data. This is the reason why we focus on the computation of rank probabilities in this chapter. Another piece of recent related work is [171], where the goal is to rank uncertain objects (i.e., x-tuples) whose score is uncertain and can be described by a range of values. Based on these ranges, the authors define a graph that captures the partial orders among objects. This graph is then processed to compute U-kRanks and other queries. Although this work has similar objectives to ours, it operates on a different input, where the distribution of uncertain scores is already known, as opposed to our work which dynamically computes this distribution by performing a linear scan over the ordered object instances.

Another efficient solution for probabilistic ranking in uncertain data has been proposed by Li et. al ([125]) at VLDB 2009 receiving the best paper award. This approach is very

similar to the approach presented in this chapter. In fact, the main difference is that [125] uses the generating functions technique (c.f. Section 3.5), while this chapter uses the Poisson binomial recurrence to compute the matching between ranking position and probability. Please note that at time of submission our the publication this chapter is based on, the publication of Li et. al had not yet been published. In fact, the work of Li et. al was published in August 2009, while our journal manuscript ([25]) was submitted April 2009; revised August 2009; accepted September 2009, but not published until April 2010. Both publications have been made independently of each other.

7.3 Probabilistic Ranking Framework

Our framework basically consists of two modules which are performed in an iterative way:

- The first module (*distance browsing*) incrementally retrieves the instances of all objects in order of their distance to q. This can be achieved with the help of a multidimensional index (e.g., an R^*-tree index [15]), using an incremental nearest neighbor search algorithm [88].

- The second module computes the probabilistic ranking $P_i(x)$ of each object instance x reported from the distance browsing for all $1 \leq i \leq k$. In essence, this module requires to perform a probabilistic ϵ-range query (c.f. Chapter 4) at each instance reported from the distance browsing. This step is the main focus of this chapter, because of its potentially high computational cost. A naive solution can perform this computation in quadratic time and linear space [203], by running an ϵ-range query from scratch at each instance, then applying the Poisson binomial recurrence. In this chapter, we improve this method to a linear time and space complexity algorithm. The key idea is an incremental computation, that uses the probabilistic ranks of the previous object instance to derive those of the currently accessed one in $O(k)$ time. Section 7.3.2 has the details of this improvement.

Our framework is illustrated in Figure 7.2. The computation of the probability distributions is iteratively processed within a loop. First, we initialize a distance browsing among the object instances starting from a given query point q. Other orders used for the instance browsing, e.g. descending probability as discussed in [203], might possibly lead to faster algorithms if the probability distribution favors them. However, the distance based order is somewhat natural for NN search around a query point, as there exist efficient search modules that support it. Furthermore, the distance based sorting supports spatial pruning techniques in order to reduce the candidate set as far as possible due to restricted memory. For each object instance fetched from the distance browsing (Module 1), we compute the corresponding rank probabilities (Module 2) and update the rank probability distributions generated from the probabilistic ranking routine.

Finally, in a postprocessing step, the rank probability distributions computed by our framework can be used to generate a definite ranking of the objects or object instances.

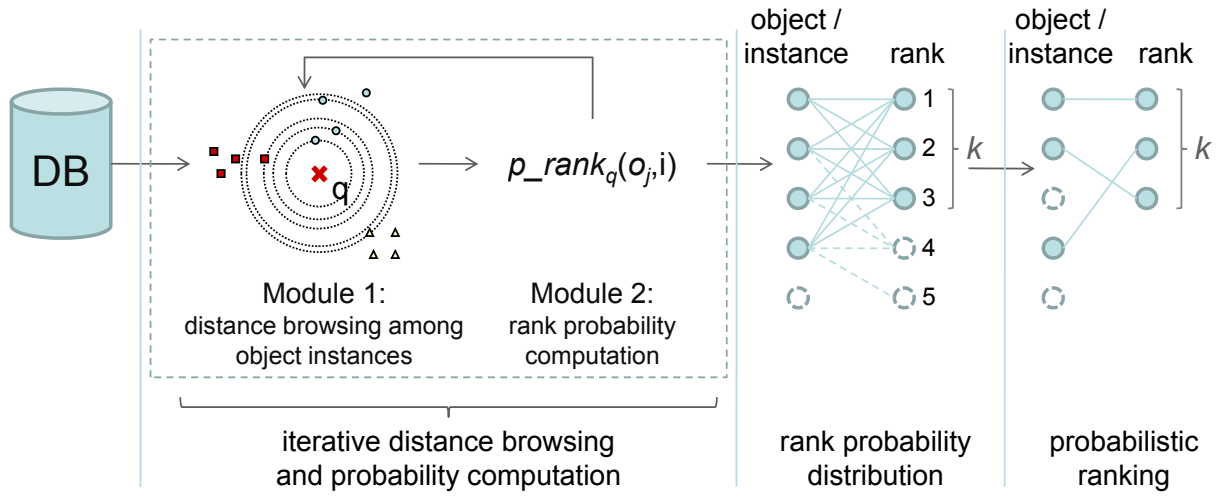


Figure 7.2: Framework for probabilistic similarity ranking.

The objective is to find a non-ambiguous ranking where each object or object instance is uniquely assigned to one rank. Here, one can plug-in any user-defined ranking method that requires rank probability distributions of objects in order to compute unique positions. In Section 7.5, we illustrate this for several well-known probabilistic ranking queries that make use of such distributions. In particular, we demonstrate that by using our framework we can process such queries in $O(nlogn + k \cdot n)$ time[1], as opposed to existing approaches that require $O(k \cdot n^2)$ time.

7.3.1 Dynamic Probability Computation

Consider an uncertain object X, defined by m probabilistic instances $X = \{(x_1, P(X = x_1)), \ldots, (x_m, P(X = x_m))\}$. The probability that X is assigned to a given ranking position i is equal to the chance that exactly $i-1$ objects $Z \in (\mathcal{DB} \setminus X)$ are closer to the query object q than the object X. This can be computed by aggregating the probabilities over all instances $(x, P(X = x))$ of X that exactly $i-1$ objects Z are closer to q than the instance $(x, P(X = x))$. Formally,

$$P_i(X) = \sum_{(x,P(X=x))\in X} (P_i(x) \cdot P(X = x)). \tag{7.1}$$

Based on the above formula we can compute the probabilities for an object X to be assigned to each of the ranking positions $i \in \{1, \ldots, k\}$ by computing the probabilities $P_i(x)$ for all instances $(x, P(X = x))$ of X. As mentioned above, we perform this computation in

[1]Note that the $O(nlogn)$ factor is due to pre-sorting the object instances according to their distances to the query object. If we assume that the instances are already sorted then our framework can compute the probability distributions for the first k rank positions in $O(k \cdot n)$ time.

Table of Notations	
$\mathcal{DB}$	an uncertain database
N	the cardinality of $\mathcal{DB}$
q	a query vector in respect to which a probabilistic ranking is computed
k	the ranking depth that determines the number of ranking positions of the ranking query result
D	a distance browsing of $\mathcal{DB}$ with respect to q
X,Y,Z	uncertain vector objects, each corresponding to a finite set of alternative vector point instances
x,y,z	vector point instances belonging to objects X,Y,Z respectively.
$P(X = x)$	the probability that an uncertain vector object X matches a given vector point instance x.
$P_i(X)$	the probability that object X is assigned to the i-th ranking position i, i.e. the probability that exactly (i-1) objects in $(\mathcal{DB} \setminus \{X\})$ are closer to q than X
$P_i(x)$	the probability that an instance x of object X is assigned to the i-th ranking position i, i.e. the probability that exactly $i-1$ objects in $(\mathcal{DB} \setminus \{X\})$ are closer to q than x
AOL	*Active Object List*
$\mathcal{S}$	a set of objects that have already been seen, i.e. the set that contains an object X *iff* at least one instance of X has already been returned by the distance browsing D
$P_{i,S,x}$	the probability that exactly i objects $X \in S$ are closer to q than an object instance x
$P_x(Z)$	the probability that object Z is closer to query point q than the vector point x; computable using Lemma 24

Table 7.1: Table of notations used in this chapter.

an iterative way, i.e., whenever we fetch a new object instance $(x, P(X = x))$ we compute all probabilities $P_i(x) \cdot P(X = x)$ for all $i \in \{1, \ldots, k\}$. A list stores the current *probability state* according to all ranking positions $i \in \{1, \ldots, k\}$ for each object for which we already have accessed some instances and for which we expect to obtain further instances in the remaining iterations. Whenever the probabilities according to a new object instance are computed, we update the list by adding the new probabilities to the current probability state.

In the following, we show how to compute the probabilities $P_i(x) \cdot P(X = x)$ for all $i \in \{1, \ldots, k\}$ for a given object instance $(x, P(X = x))$ of an uncertain object X which is assumed to be currently fetched from the distance browsing (Step 1). For this computation we first need, for all uncertain objects $Z \in \mathcal{DB}$, the probability $P_x(Z)$ that Z is closer to q than the current object instance x. These probabilities are stored in an *active object list* AOL, which can easily be kept updated due to the following obvious lemma:

Lemma 24. *Let q be the query object and $(x, p(X = x))$ be the object instance of an object X fetched from the distance browsing in the current processing iteration. The probability that an object $Z \neq X$ is closer to q than x is*

$$P_x(Z) = \sum_{(z,P(Z=z))\in Z} P(Z = z),$$

where $(z, P(Z = z))$ are the instances fetched in previous processing iterations.

Lemma 24 says that we can accumulate in overall linear space the sums of probabilities of all instances for each object, which have been seen so far and use them to compute $P_x(Z)$ given the current instance x and any object Z in $\mathcal{D}$. In fact, we only need to manage in the list the probabilities of those objects for which we already have accessed an instance and for which we expect to access further instances in the remaining iterations.

Now let us see how we can use list AOL to efficiently compute the probabilities $P_i(x)$. Assume that $(x, P(X = x)) \in X$ is the current object instance reported from distance browsing. Let $\mathcal{S} = \{Z_1, \ldots, Z_j\}$ be the set of objects which have been seen so far, i.e. for which we already have seen at least one object instance. The probability that an object $X \in \mathcal{S}$ appears at ranking position i of the first j objects seen so far only depends on the event that $i - 1$ of the remaining $j - 1$ objects $Z \in \mathcal{S}$ $(Z \neq X)$ appear before X, no matter which of these objects fulfill this criterion. Let $\mathcal{S}$ denote the set of objects except for object X seen so far, i.e. $X \notin \mathcal{S}$. Furthermore, let $P_{i,\mathcal{S},x}$ denote the probability that exactly i objects of $\mathcal{S}$ are closer to q than the object instance x. Now, we can exploit the Poisson binomial recurrence (c.f. Section 3.4):

$$P_{i,\mathcal{S},x} = P_{i-1,\mathcal{S}\setminus\{Z\},x} \cdot P_x(Z) + P_{i,\mathcal{S}\setminus\{Z\},x} \cdot (1 - P_x(Z)),$$

where

$$P_{0,\emptyset,x} = 1 \text{ and } P_{i,\mathcal{S},x} = 0, \text{ if } i > |\mathcal{S}| \vee i < 0. \tag{7.2}$$

The correctness of Equation 7.2 can be shown using the intuition of the Poisson binomial recurrence: the event that i objects of $\mathcal{S}$ are closer to q than x occurs if one of the following conditions holds. In the case that an object $Z \in \mathcal{S}$ is closer to q than x, then $i - 1$ objects of $\mathcal{S} \setminus \{Z\}$ must be closer to q. Otherwise, if we assume that object $Z \in \mathcal{S}$ is farther to q than x, then i objects of $\mathcal{S} \setminus \{X\}$ must be closer to q.

For each object instance $(x, P(X = x))$ reported from the distance browsing, we have to apply the recursive function as defined above. Specifically, we have to compute for each instance $(x, P(X = x))$ the probabilities $P_{i,\mathcal{S},x}$ for all $i \in \{0, \ldots, \min\{k, |\mathcal{S}|\}\}$ and for $j = |\mathcal{S}|$ subsets of $\mathcal{S}$. If $n = |\mathcal{DB}|$, this has a cost factor of $\mathrm{O}(k \cdot n)$ per object instance retrieved from the distance browsing, leading to a total cost of $\mathrm{O}(k \cdot n^2)$. Assuming that k is a small constant, we have an overall runtime of $\mathrm{O}(n^2)$.

In the following, we show how we can compute each $P_{i,\mathcal{S},x}$ in constant time by utilizing the probabilities computed for the previously accessed instance.

7.3.2 Incremental Probability Computation

Let $(x, P(X = x)) \in X$ and $(y, P(Y = y)) \in Y$ be two object instances consecutively returned from the distance browsing. W.l.o.g. let $(x, P(X = x))$ be returned before $(y, P(Y = y))$. Each of the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ ($i \in \{0, \ldots, |\mathcal{S} \setminus \{Y\}|\}$) can be computed from the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ in constant time. In fact, the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ can be computed by considering at most one recursion step backward.

The following three cases have to be considered. The first two are easy to tackle and the third one is the most common and challenging one.

Case 1: Both instances belong to the same object, i.e. $X = Y$.

Case 2: Both instances belong to different objects, i.e. $X \neq Y$ and $(y, P(Y = y))$ is the first returned instance of object Y.

Case 3: Both instances belong to different objects, i.e. $X \neq Y$ and $(y, P(Y = y))$ is not the first returned instance of object Y.

Now, we show how the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ for $i \in \{0, \ldots, |\mathcal{S}\setminus\{Y\}|\}$ can be computed in constant time considering the above cases which are illustrated in Figure 7.3.

In the first case (cf. Figure 7.3(a)), the probabilities $P_x(Z)$ and $P_y(Z)$ of all objects in $Z \in \mathcal{S} \setminus \{X\}$ are equal, because the instances of objects in $\mathcal{S} \setminus \{X\}$ that appear within the distance range of q of y and within the distance range of x are identical. Since the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ and $P_{i,\mathcal{S}\setminus\{X\},x}$ only depend on $P_x(Z)$ for all objects $Z \in \mathcal{S} \setminus \{X\}$, it is obvious that $P_{i,\mathcal{S}\setminus\{Y\},y} = P_{i,\mathcal{S}\setminus\{X\},x}$ for all i.

In the second case (cf. Figure 7.3(b)) we can exploit the fact that $P_{i,\mathcal{S}\setminus\{X\},x}$ does not depend on Y. Thus, given the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$, we can easily compute the probability $P_{i,\mathcal{S}\setminus\{Y\},y}$ by incorporating the object X using the recursive Equation 7.2:

$$P_{i,\mathcal{S}\setminus\{Y\},y} =$$

$$P_{i-1,\mathcal{S}\setminus\{Y,X\},y} \cdot P_y(X) + P_{i,\mathcal{S}\setminus\{Y,X\},y} \cdot (1 - P_y(X)).$$

Since $\mathcal{S} \setminus \{Y, X\} = \mathcal{S} \setminus \{X, Y\}$ and no instance of any object in $\mathcal{S} \setminus \{X, Y\}$ appears within the distance range of q according to y but not within the range according to x (cf. Figure 7.3(b)), the following equation holds:

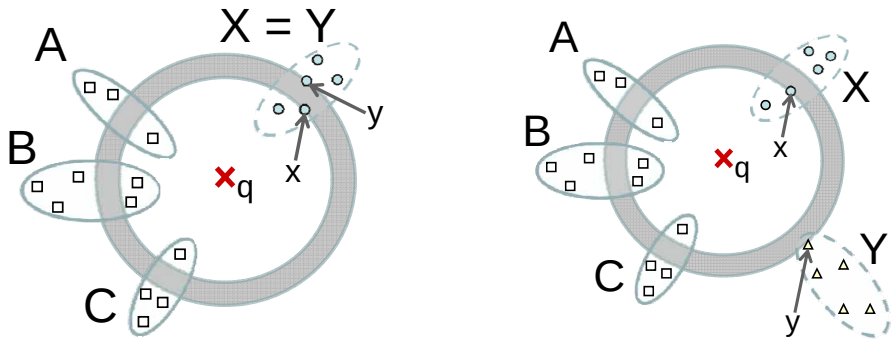
$$P_{i,\mathcal{S}\setminus\{Y\},y} =$$

$$P_{i-1,\mathcal{S}\setminus\{X,Y\},x} \cdot P_y(X) + P_{i,\mathcal{S}\setminus\{X,Y\},x} \cdot (1 - P_y(X)).$$

Furthermore, $P_{i-1,\mathcal{S}\setminus\{X,Y\},x} = P_{i-1,\mathcal{S}\setminus\{X\},x}$, because Y is not in the distance range according to x and, thus, $Y \notin \mathcal{S} \setminus \{X\}$. Now, the above equation can be reformulated:

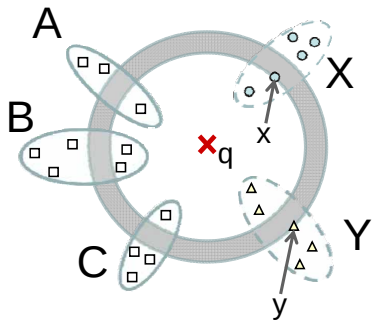
$$P_{i,\mathcal{S}\setminus\{Y\},y} =$$

$$P_{i-1,\mathcal{S}\setminus\{X\},x} \cdot P_y(X) + P_{i,\mathcal{S}\setminus\{X\},x} \cdot (1 - P_y(X)). \tag{7.3}$$



(a) Case 1: Previous instance x and current instance y belong to the same object.

(b) Case 2: Instance y is the first returned instance of object Y.



(c) Case 3: Instance y is not the first returned instance of object Y and $X \neq Y$.

Figure 7.3: Cases when updating the probabilities, assuming x was the last processed instance and y is the current one.

All probabilities of the term on the right hand side in Equation 7.3 are known and, thus, $P_{i,\mathcal{S}\setminus\{Y\},y}$ can be computed in constant time assuming that the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ computed in the previous step have been stored for all $i \in \{0,\ldots,|\mathcal{S}\setminus\{X\}|\}$.

The third case (cf. Figure 7.3(c)) is the general case which is not as straightforward as the previous two cases and requires special techniques. Again, we assume that the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ computed in the previous step for all $i \in \{0,\ldots,|\mathcal{S}\setminus\{X\}|\}$ are known. Similar to Case 2, the probability $P_{i,\mathcal{S}\setminus\{Y\},y}$ is equal to:

$$P_{i,\mathcal{S}\setminus\{Y\},y} =$$

$$P_{i-1,\mathcal{S}\setminus\{X,Y\},x} \cdot P_y(X) + P_{i,\mathcal{S}\setminus\{X,Y\},x} \cdot (1 - P_y(X)). \tag{7.4}$$

Since the probability $P_y(X)$ is assumed to be known, now we are left with the computation of $P_{i,\mathcal{S}\setminus\{X,Y\},x}$ for all $i \in \{0,\ldots,|\mathcal{S}\setminus\{X,Y\}|\}$ by again exploiting Equation 7.2:

$$P_{i,\mathcal{S}\setminus\{X\},x} =$$

$$P_{i-1,\mathcal{S}\setminus\{X,Y\},x} \cdot P_x(Y) + P_{i,\mathcal{S}\setminus\{X,Y\},x} \cdot (1 - P_x(Y))$$

which can be resolved to

$$P_{i,\mathcal{S}\setminus\{X,Y\},x} =$$

$$\frac{P_{i,\mathcal{S}\setminus\{X\},x} - P_{i-1,\mathcal{S}\setminus\{X,Y\},x} \cdot P_x(Y)}{1 - P_x(Y)}. \tag{7.5}$$

With $i = 0$ we have

$$P_{0,\mathcal{S}\setminus\{X,Y\},x} = \frac{P_{0,\mathcal{S}\setminus\{X\},x} - P_{-1,\mathcal{S}\setminus\{X,Y\},x} \cdot P_x(Y)}{1 - P_x(Y)} =$$

$$\frac{P_{0,\mathcal{S}\setminus\{X\},x}}{1 - P_x(Y)},$$

because the probability $P_{-1,\mathcal{S}\setminus\{X,Y\},x} = 0$ by definition (cf. Equation 7.2). The case $i = 0$ can be solved assuming that $P_{0,\mathcal{S}\setminus\{X\},x}$ is known from the previous iteration step.

With the assumption that all probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ for all $i \in \{1, \ldots, |\mathcal{S} \setminus \{X\}|\}$ and $P_x(Y)$ are available from the previous iteration step, we can use Equation 7.5 to recursively compute $P_{i,\mathcal{S}\setminus\{X,Y\},x}$ $(1 \leq i \leq |\mathcal{S} \setminus \{X, Y\}|)$ using the previously computed $P_{i-1,\mathcal{S}\setminus\{X,Y\},x}$. Based on this recursive computation we obtain all probabilities $P_{i,\mathcal{S}\setminus\{X,Y\},x}$ $(0 \leq i \leq |\mathcal{S} \setminus \{X, Y\}|)$ which can used to compute the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ for all $0 \leq i \leq |\mathcal{S} \setminus \{X, Y\}|$ according to Equation 7.4.

7.3.3 Runtime Analysis

Building on this case-based analysis for the cost of computing $P_{i,\mathcal{S}\setminus\{X\},x}$ for the currently accessed instance x of an object X, we now prove that we can compute the rank probabilities of all objects at cost $O(nk)$, where n is the number of object instances accessed. The following lemma suggests that the incremental cost per object instance access is $O(k)$.

Lemma 25. *Let $(x, P(X = x)) \in X$ and $(y, P(Y = y)) \in Y$ be two object instances consecutively returned from the distance browsing. W.l.o.g., let us assume that the instance $(x, P(X = x))$ was returned in the last iteration in which we computed the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ for all $0 \leq i \leq |\mathcal{S} \setminus \{X\}|$. The next iteration, in which we fetch $(y, P(Y = y))$ the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ for all $0 \leq i \leq \min\{k, |\mathcal{S} \setminus \{Y\}|\}$, can be computed in $O(k)$ time and space.*

Proof. In Case 1, the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ and $P_{i,\mathcal{S}\setminus\{Y\},y}$ are equal for all $0 \leq i \leq min\{k, |\mathcal{S} \setminus \{Y\})|\}$. No computation is required ($O(1)$ time) and the result can be stored using at most $O(k)$ space.

In Case 2, the probabilities $P_{i,\mathcal{S}\setminus\{Y\},y}$ for all $0 \leq i \leq min\{k, |\mathcal{S}\setminus\{Y\})|\}$ can be computed according to Equation 7.3 taking $O(k)$ time. This assumes that the $P_{i,\mathcal{S}\setminus\{X\},x}$ have to be stored for all $0 \leq i \leq \min\{k, |\mathcal{S} \setminus \{Y\}|\}$, requiring at most $O(k)$ space.

runtime table	no precomputed D	precomputed D
ours	O($n$log$n$+kn)	O(kn)
[125]	O($n$log$n$+kn)	O(kn)
[202, 203]	O(kn^2)	O(kn^2)
[27]	exponential	exponential
[172]	exponential	exponential
[129]	exponential	exponential

Table 7.2: Runtime complexity comparison of the best-known approaches to our own approach.

In Case 3, we first have to compute and store the probabilities $P_{i,\mathcal{S}\setminus\{X,Y\},x}$ for all $0 \leq i \leq \min\{k, |\mathcal{S} \setminus \{X,Y\})|\}$ using the recursive function in Equation 7.5. This can be done in $O(\min\{k, |\mathcal{S} \setminus \{X,Y\})|\})$ time and space. Next, the computed probabilities can be used to compute $P_{i,\mathcal{S}\setminus\{Y\},y}$ for all $0 \leq i \leq \min\{k, |\mathcal{S} \setminus \{Y\})|\}$ according to Equation 7.4 which again takes at most $O(\min\{k, |\mathcal{S} \setminus \{X,Y\})|\})$ time and space. □

After giving the runtime evaluation of the processing of one single object instance, we are now able to extend the cost model for the whole query process. According to Lemma 25, we can assume that each object instance can be processed in constant time if we assume that k is constant. If we assume that the total number of object instances in our database is linear to the number of database objects we would get a runtime complexity which is linear in the number of database objects, more exactly particular O(kn) where n is the size of the database and k the specified depth of the ranking. Up to now, our model assumes that the preprocessing step and the postprocessing step of our framework requires at most linear runtime. Since the postprocessing step only includes an aggregation of the results generated in Step 2 the linear runtime complexity of Step 3 is guaranteed. Now, we want to examine the runtime of the object instance ranking in Step 1. Similar to the assumptions that hold for our competitors [172, 202, 27] we can also assume that the object instances are already sorted, which would involve linear runtime cost also for Step 1. However, for the general case where we have to initialize a distance browsing first, the runtime complexity of Step 1 would increase to O($n$log$n$). As a consequence, the total runtime cost of our approach (including distance browsing) sums up to O($n$log$n$+kn). An overview of the computation cost is given in Table 7.2. Keep in mind that the approach proposed in [172] uses result based answer semantics, so for this work, the comparison is not entirely fair. Also, recall that the approach of [55] has been published concurrently and independently of the approach presented in this chapter.

Regarding the space complexity of our approach, we have to store, for each object in the database, a vector of length k for the probabilistic ranking of size $O(kn)$. In addition, we have to store the AOL of at most size $O(n)$, yielding a total space complexity of $O(kn + n) = O(kn)$. Note that [202, 203] computes a different ranking (cf. Section 7.5 for details) with a space complexity of $O(n)$. To compute a probabilistic ranking according to our definition, [202, 203] requires $O(kn)$ space as well.

7.4 Probabilistic Ranking Algorithm

Algorithm 2 Pseudocode of our ranking algorithm.

Probabilistic Ranking($\mathcal{DB}$,q)

Input: Database $\mathcal{DB}$, Query Vector q

```
1   AOL = ∅
2   result = Matrix of zeros // size: |instances|*k
3   P_i(x) = [0,...,0] // Length k
4   P_i(y) = [0,...,0] // Length k
5
6   y = DB.next
7   updateAOL(y)
8   P_i(x)[0]=1
9   Add P_i(x) to the first line of result.
10  FOR (DB is not empty AND ∃p ∈ P_i(x): p > 0)
11    x = y
12    y = DB.next
13    updateAOL(y)
14
15    CASE 1: (c.f. Figure 7.3(a))
16    IF (Y = X)
17      P_i(y) = P_i(x)
18    END-IF
19
20    CASE 2: (c.f. Figure 7.3(b))
21    ELS-IF (Y ∉ AOL)
22      P(X)=AOL.getProb(X)
23      P_i(y) = dynamicRound(P_i(x),P_y(X))])
24    END-IF
25
26    CASE 3: (c.f. Figure 7.3(c))
27    ELSE            // (Y != X)
28      P(X)=AOL.getProb(X)
29      P(Y)=AOL.getProb(Y)
30      adjustedProbs = adjustProbs(P_i(x),P_y(Y))
31      p-rank_y = dynamicRound(adjustedProbs,P_y(X))
32    END-IF
33
34    Add P_i(y) to the next line of result.
35    P_i(x) = P_i(y)
36  END-FOR
37  return result
38  END Probabilistic Ranking.
```

Output: Probabilistic Ranking (c.f. Definition)

The pseudocode of the algorithm for the probabilistic ranking is illustrated in Algorithm 2, providing the implementation details of the previously discussed steps. Our algorithm requires a query object q and a distance browsing operator D (cf. [88]), that allows us to

Algorithm 3 Pseudocode of a dynamic Iteration at instance y

dynamicRound(*oldRanking*,$P_y(X)$)

Input: *oldRanking*: Intermediate result without object X
$P_y(X)$: Prob. that object X is closer to q than instance y.

```
1   newRanking = [0,...,0] // Length k
2   newRanking[0] =
3      oldRanking[0]*(1-P_y(X))
4   FOR i = 1,...,k-1
5     newRanking[i] =
6        oldRanking[i-1]*P_y(X)
7        +oldRanking[i]*(1-P_y(X))
8   END-FOR
9    return newRanking
10  END dynamicRound.
```

Output: Result including object X

Algorithm 4 Pseudocode of the algorithm that excludes one object Y from the current result at instance $y \in Y$.

adjustProbs(*oldRanking*,$P_y(Y)$)
Input: *oldRanking*: Intermediate result including object Y
$P_y(X)$: Prob. that another instance of object Y is closer to q than instance y.

```
1   adjustedRanking = [0,...,0] // Length k
2   adjustedProbs[0] =
3   oldRanking[0] / P_y(Y)
4   FOR i = 1,...,k-1
5     adjustedProbs[i] = (oldRanking[i]-oldRanking[i-1]*P_y(Y)) / (1-P_y(Y))
6   END-FOR
7   return adjustedProbs
8   END adjustProbs.
```

Output: Intermediate result at instace y excluding object Y

iteratively access the object instances sorted in ascending order of their similarity distance to a query object.

First, we initialize the *active Object List* (AOL) , a data structure that contains one tuple $(X, P(X))$ for each object X that

- has previously been found in D, i.e. at least one instance of X has been processed and
- has not yet been completely processed, i.e. at least one instance of X has yet to be found,

associated with the sum $P(X)$ of probabilities of all its instances that have been found. The AOL offers two functionalities:

- updateAOL(instance x): Adds the probability of x ($P(X = x)$) to $P(X)$, where X is the object that x belongs to.
- getProb(object X): Returns $P(X)$.

Note that it is mandatory that the position of a tuple $(X, P(X))$ can be found in constant time, in order to sustain the constant time complexity of an iteration. This can be

- approached by means of hashing or
- reached by giving each object X the information about the location of its corresponding instances ($P(X)$) at an additional space cost of $O(n)$.

We also keep the *result*, a matrix that contains, for each object instance x that has been found and each ranking position i, the probability $P_i(x)$ that x is located at ranking position i. Note that this result is instance-based. In order to get an object-based rank probability, we can aggregate instances belonging to the same object, using Equation 7.1. Additionally, we initialize two arrays *p-rank*_x and *p-rank*_y, each of length k, which contain, at any iteration of the algorithm, the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ and $P_{i,\mathcal{S}\setminus\{Y\},y}$ respectively, for all $0 \leq i \leq k$. $x \in X$ is the instance found in the previous iteration and $y \in Y$ is the instance found in the current iteration (see Figure 7.3).

In line 6, the algorithm starts by fetching the first object instance, which is closest to the query q in the database. A tuple containing the corresponding object as well as the probability of this instance is added to the AOL.

Then, the first position of *p-rank*_x is set to 1 while all other $k-1$ positions remain at 0, because

$$P_{1,\mathcal{S}\setminus\{y\},y} = P_{1,\emptyset,y} = 1$$

and

$$P_{i,\mathcal{S}\setminus\{y\},y} = P_{i,\emptyset,y} = 0$$

for $i > 1$ by definition (see Equation 7.2). This simply reflects the fact that the first instance is always on rank 1. Note that *p-rank*_y is implicitly assigned to *p-rank*_x here.

Then, the first iteration of the main algorithm begins by fetching the next object instance from D. Now, we have to distinguish the three cases explained in Section 7.3.

In the first case (line 16), both the previous and the current instance refer to the same object. As explained in Section 7.3, we have nothing to do in this case, since $P_{i,\mathcal{S}\setminus\{X\},x}$=$P_{i,\mathcal{S}\setminus\{Y\},y}$ for all $0 \leq i \leq k-1$.

In the second case (line 21), the current instance refers to an object that has not been seen yet. As explained in Section 7.3, we only have to apply an additional iteration of the DP algorithm (cf. Equation 7.2). This *dynamicRound* algorithm is shown in Algorithm 3 and is used here to incorporate the probability that X is closer to y into *p-rank*_y in a single iteration of the dynamic algorithm.

In the third case (line 27), the current instance relates to an object that has already been seen. Thus the probabilities $P_{i,\mathcal{S}\setminus\{X\},x}$ depend on Y. As explained in Section 7.3, we

first have to filter out the influence of Y on $P_{i,\mathcal{S}\setminus\{X\},x}$ and compute $P_{i,\mathcal{S}\setminus\{X,Y\},x}$. This is performed by the *adjustProbs* algorithm in Algorithm 4 utilizing the technique explained in Section 7.3. Using the $P_{i,\mathcal{S}\setminus\{X,Y\},x}$, the algorithm then computes the $P_{i,\mathcal{S}\setminus\{Y\},y}$ using a single iteration of the dynamic algorithm like in case two.

At line 35, the computed ranking for instance y is added to the result. If the application (i.e. the ranking method) requires objects to be ranked instead of instances, then *p-rank*_y is used to incrementally update the probabilities of Y for each rank.

The algorithm continues fetching object instances from the distance browsing operator D and repeats this case analysis until either no more samples are left in $\mathcal{DB}$ or until an object instance is found, for the probability zero for each of the first k positions. In the later case, there exist k objects, that are closer to k with a probability of one and the computation can be stoped, because the same k objects must be closer to all further object instances in the database that have not yet been found.

7.5 Probabilistic Ranking Approaches

The method proposed in Section 7.3 efficiently computes for each uncertain object instance x_j and each ranking position i $(0 \leq i \leq k-1)$ the probability that x_j has the i^{th} rank. However, most applications require an unique object ranking, i.e. each object (or object instance) is uniquely assigned to exactly one rank. Various top-k query approaches have been proposed generating deterministic rankings from probabilistic data which we call probabilistic ranking queries. The question at issue is how our framework can be exploited in order to significantly accelerate probabilistic ranking queries. In the remainder, we show that our framework is able to support and significantly boost the performance of the state-of-the-art probabilistic ranking queries. Specifically, we demonstrate this by applying state-of-the-art ranking approaches including, U-kRanks, PT-k and *Global top-k*.

Note, that the following ranking approaches are based on the x-relation model [16, 7]. As mentioned before, the x-relation model conceptionally corresponds to our uncertainty model for uncertain spatial data, where the possible spatial locations (instances) correspond to tuples and uncertain spatial objects correspond to x-tuples.

7.5.1 Expected Score and Expected Ranks

The *Expected Score* and *Expected Ranks* [55] compute for each object instance its expected score (rank) and rank the instances by this expected score (rank). *Expected Ranks* runs in $O(n \cdot log(n))$-time, thus outperforming exact approaches that do not use any estimation. The main drawback of this approach is that by using the expected value estimator, information is lost about the distribution of the objects. In the following, we will show how our framework can be used to accelerate the remaining state-of-the-art approaches, including U-kRanks, PT-k and *Global top-k*, to $O(n \cdot logn + kn)$ runtime.

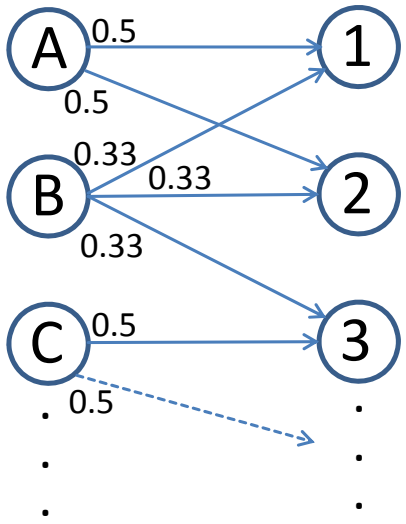


Figure 7.4: Small example extract of a probabilistic ranking as produced by our framework.

7.5.2 U-kRanks

The U-kRanks [172] approach reports the most likely object instance at each rank i, i.e. the instance that is most likely to be ranked i-th over all possible worlds. The approach proposed in [172] has exponential runtime. The runtime has been reduced to $O(n^2k)$ time in [202, 203]. Using our framework, the problem of U-kRanks can be solved in $O(n \cdot log(n) + nk)$ time using the same space complexity as follows:
Use the framework to create the probabilistic ranking in $O(n{\cdot}log(n){+}nk)$ as explained in the previous section. Then, for each rank i, find the object instance $argmax_j(p\_rank_q(X_j, i))$ that has the highest probability of appearing at rank i in $O(nk)$. This is performed by (cf. Figure 7.4) finding for each rank i the object instance which has the highest probability to be assigned to rank i. Obviously, a problem of this problem definition is that a single object instance o_j may appear at more than one ranking position, or at no ranking position at all. For example in 7.4, object instance A is ranked on both ranks 1 and 2, while object instance B is ranked nowhere. The total runtime for U-kRanks has thus been reduced from $O(n^2)$ to $O(nlog(n) + kn)$, that is $O(n * log(n))$ if k is assumed to be constant.

7.5.3 PT-k

The *probabilistic threshold top-k* query (PT-k) [91] problem fixes the problem of the previous definition by aggregating the probabilities of an object instance x_j appearing at rank k or better. Given a user-specified probability threshold p, PT-k returns all instances, that have a probability of at least p of being at rank k or better. Note that in this definition, the number of results is not limited to k and depends on the threshold parameter p. The model of PT-k consists of a set of instances and a set of generation rules that define mutually exclusiveness of instances. Each object instance occurs in one and only

one generation rule. This model conceptionally corresponds to the x-relation model (with disjoint x-tupels). PT-k computes all result instances in $O(nk)$ time while also assuming that the instances are already pre-sorted, thus having a total runtime of $O(nlog(n) + kn)$. The framework can be used to solve the PT-k problem in the following way:
We create the probabilistic ranking in $O(nk)$ as explained in the previous section. For each object instance x, we compute the probability that x appears at position k or better (in $O(nk)$). Formally, we return all instances $x \in \mathcal{DB}$ for which:

$$\{x \in \mathcal{DB} | \sum_{i=1}^{k} P_i(x) > p\}$$

As seen in Figure 7.4, this probability can simply be computed by aggregating all probabilities of an object instance to be ranked at k or better. For example, for $k = 2$ and $p = 0.5$, we get A and B as results. Note that for $p = 0.1$, further object instances may be in the result, because there must be further object instances (from object instances that are left out here for simplicity) with a probability greater than zero to rank 1 and rank 2, since the probability of their respective edges does not sum up to 1.0 yet.

Note that our framework is only able to match, not to beat the runtime of PT-k. However, using our approach, we can additionally return the ranking order, instead of just the top-k set.

7.5.4 Global top-k

Global top-k [213] is very similar to PT-k and ranks the object instances by their top-k probability, and then takes the top-k of these. This approach has a runtime of $O(n^2k)$. The advantage here is that, unlike in PT-k, the number of results is fixed, and there is no user-specified threshold parameter. Here we can exploit the ranking order information that we acquired in the PT-k using our framework to solve *Global top-k* in $O(n \cdot log(n) + kn)$ time:
We use the framework to create the probabilistic ranking in $O(n \cdot log(n) + kn)$ as explained in the previous section. For each object instance x, we compute the probability that x appears at position k or better (in $O(nk)$) like in PT-k. Then, we find the k object instances with the highest probability in $O(k \cdot log(k))$.

7.6 Experimental Evaluation

We have performed extensive experiments to evaluate the performance of our proposed probabilistic ranking approach proposed in Section 7.3 w.r.t. the database size ($|\mathcal{DB}|$) measured in the number of uncertain vector objects, ranking depth (k) and degree of uncertainty (UD) as defined below. In the following, the ranking framework is briefly denoted by **PSR**.

7.6.1 Datasets and Experimental Setup

The probabilistic ranking was applied to a scientific real-world dataset *SCI* and several artificial datasets *ART_X* of varying size and degree of uncertainty. All datasets are based on the discrete uncertainty model, i.e. each object is represented by a collection of vector samples.

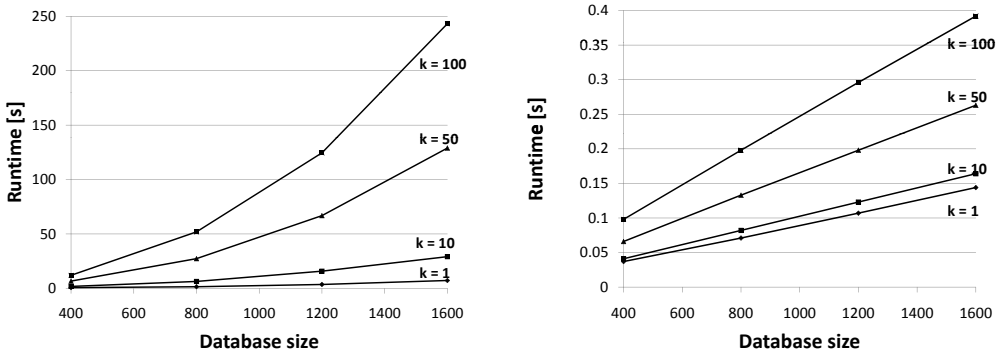
The *SCI2* dataset that has been presented in Section 4.6.3 consisting of 1500 objects where each object consists of 48 10-dimensional instances and each dimension represents the concentrations of gasses such as CO, SO_2, NO, NO_2, and O_3 measured over time. These attributes are normalized within the interval [0,1] to give each attribute the same weight.

The *ART_1* dataset consists of 1,000,000 objects, each consisting of 20 object instances for the scalability experiments. For the evaluation of the performance w.r.t. the ranking depth and the degree of uncertainty we applied a collection *ART_2* of datasets each composing 10,000 objects. Each object is represented by a set of 20 3-dimensional instances. The *ART_2* datasets differs in the degree of uncertainty (UD) the corresponding objects have. The degree of uncertainty (UD) reflects the following distribution of object instances: each uncertain vector object is assumed to be located within an 3-dimensional hyper-rectangle. The object instances are uniformly distributed within the corresponding rectangle. In the following, we will refer to the side length of the rectangles as *degree of uncertainty* (UD). The rectangles are uniformly distributed within a $10 \times 10 \times 10$ vector space. The *ART_3* datasets are very similar to *ART_3* datasets, except that the instances of object (again 10,000 objects uniformly distributed in the vector space with 20 instances each) follow a three dimensional normal distribution. The datasets of *ART_3* vary in the degree of uncertain as well. For this dataset, the degree of uncertain simply denotes the standard deviation of the normal distribution of the objects.

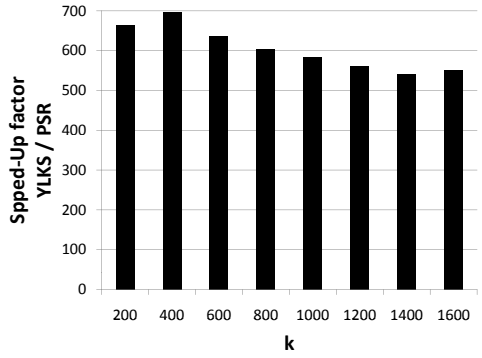
The degree of uncertainty is interesting in our performance evaluation since it is expected to have a significant influence on the runtime. The reason is that a higher degree of uncertainty obviously leads to an higher overlap between the objects which influences the size of the active object list (AOL) (cf. Section 7.4) during the distance browsing. The higher the object overlap the more objects are expected to be in the AOL at a time. Since the size of the AOL influences the runtime of the rank probability computation, a higher degree of uncertainty is expected to lead to a higher runtime. This is experimentally evaluated in Section 7.6.4.

7.6.2 Scalability

In this section, we give an overview of our experiments regarding the scalability of **PSR**. We compare our results to the dynamic programming based rank probability computation used for the U-kRanks method as proposed by Yi et al. in [202]. This method, in the following denoted by **YLKS**, is the best approach currently known for solving the (instance-based) rank probability problem (cf. Table 7.2). For a fair comparison, we used the **PSR** framework to compute the same (instance-based) rank probability problem as described in



(a) **YLKS** (b) **PSR**



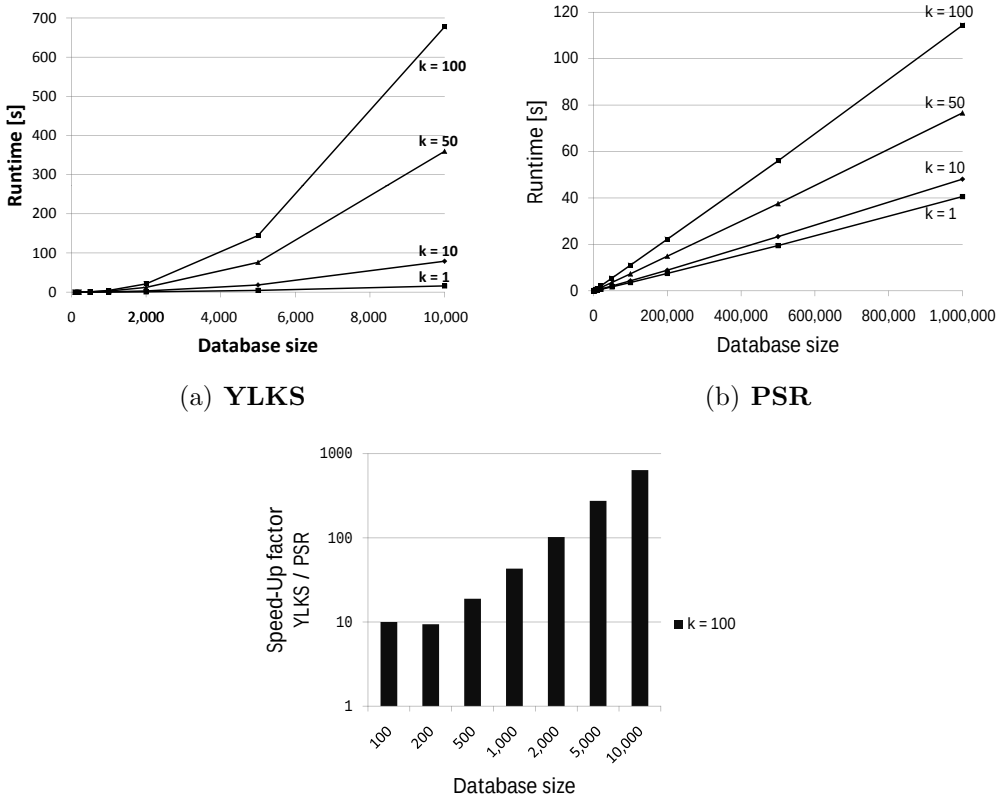
(c) Speed-up gain w.r.t. k on *SCI2*.

Figure 7.5: Scalability evaluated on *SCI2* for different k values.

Section 7.3. Let us note that the cost required to solve the object-based rank probability problem is similar to that required to solve the instance-based rank probability problem. This is because the former problem additionally only requires to build the sum over all instance-based rank probabilities which can be done on-the-fly without additional cost. Furthermore, we can neglect the cost required to build a final definite ranking (e.g. the rankings proposed in Section 7.5) from the rank probabilities, because they can be also computed on-the-fly by simple aggregations of the corresponding (instance-based) rank probabilities.

For the sorting of the distances of the instances to the query point, we used a tuned quicksort adapted from [18]. This algorithm offers $O(n \cdot log(n))$ performance on many data sets that cause other quicksort algorithms to degrade to quadratic runtime.

The results of our first scalability tests on the real-data set *SCI2* are depicted in Figure 7.5. It can be observed in Figure 7.5(b) that the runtime of the probabilistic ranking using the **PSR** framework increases linearly in the database size, whereas **YLKS** has a runtime quadratic in the database size in the same parameter settings (cf. Figure 7.5(a)). We can also see that this effect persists for different settings of k. Note that the effect of the $O(n \cdot log(n))$ sorting of the distances of the instances is insignificant on this relatively



(a) **YLKS**

(b) **PSR**

(c) Speed-up factor

Figure 7.6: Scalability evaluated on *ART_1* for different k values.

small dataset. The direct speed-up of the rank probability computation using **PSR** in comparison to **YLKS** is depicted in Figure 7.5(c). It shows for different values of k, the speed-up factor, that is defined as the ratio $\frac{runtime(\mathbf{YLKS})}{runtime(\mathbf{PSR})}$ describing the performance gain of **PSR** vs. **YLKS**. It can be observed that, for a constant number of objects in the database ($|DB| = 1600$), the ranking depth k has no impact on the speed-up factor. This can be explained by the observation that both approaches scale linear in k.

Next, we evaluate the scalability of the database size based on the *ART_1* dataset. The results of this experiment are depicted in Figure 7.6. Figure 7.6(b) shows that we are able to perform ranking queries in a reasonable time of less than 120 seconds, even for very large database containing 1,000,000 and more objects, each having 20 instances (thus having a total of 20,000,000 instances (tuples)). Note that an almost perfect linear scaleup can be seen in Figure 7.6 despite of the $O(n \cdot log(n))$ cost for sorting the database. This is due to the very efficient quicksort implementation in [18] that our experiments have shown to require only slightly worse than linear time.

In Figure 7.6(a), it can be observed, that due to the quadratic scaling of the **YLKS** algorithm, it is inapplicable for relatively small databases of size 5000 or more. The direct speed-up of the rank probability computation using **PSR** in comparison to **YLKS** for

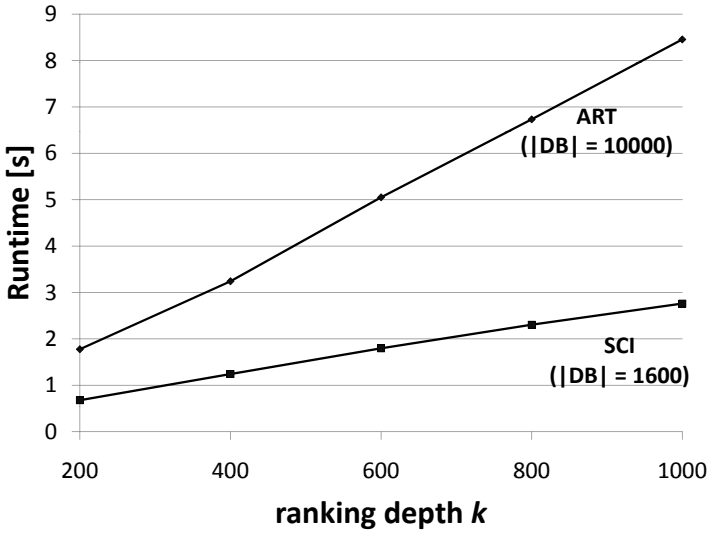


Figure 7.7: Runtime using **PSR** on *SCI2* and *ART*.

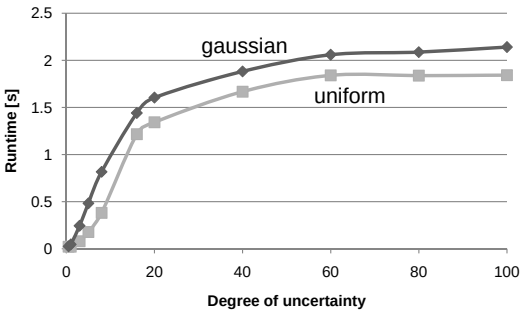
varying database size is depicted in Figure 7.6(c). Here, we can see that the speed-up of our approach in comparison to **YLKS** increases linear with the size of the database which is consistent with our runtime analysis in Section 7.3.

7.6.3 Ranking Depth k

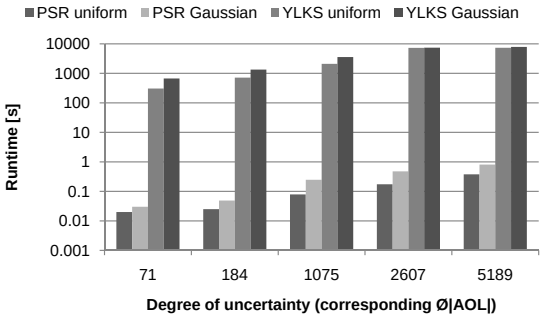
The influence of the ranking depth k on the runtime performance of our probabilistic ranking method **PSR** is studied in the next experiment. As depicted in Figure 7.7, where the experiments were performed using both the *SCI2* and the *ART* dataset, the influence of an increasing k yields a linear effect on the runtime of **PSR**, but does not depend on the type of the dataset. This effect can be explained by taking into consideration that each iteration of Case 2 or Case 3 requires a probability computation for each ranking position $0 \leq i \leq k$.

7.6.4 Influence of the Degree of Uncertainty

In the next experiment, we varied the uncertainty degree of objects using the *ART_2* and *ART_3* datasets. In the following experiments, the ranking depth is set to a fixed value of $k = 100$. As previously discussed, a varying degree of uncertainty leads to an increase of the overlap between the instances of the objects and thus, objects will remain in the *AOL* for a longer time. The influence of the degree of uncertainty depends on the probabilistic ranking algorithm. This statement is underlined by the experiments shown in Figure 7.8. It can be seen in Figure 7.8(a) that **PSR** scales superlinear in the degree of uncertainty at first, until a maximal value is reached. This maximal value is reached, when the degree of uncertainty becomes so large that the instances of an object cover the whole



(a) Evaluation of **PSR** by an increasing uncertainty degree.



(b) **YLKS** vs. **PSR** in a logarithmic scale w.r.t. different $\varnothing(|AOL|)$ values.

Figure 7.8: Runtime w.r.t. the degree of uncertainty.

vector space. In this case, objects remain on the AOL until almost the whole database is processed in most cases due to the increased overlap of object instances. In this case of extremely high uncertainty, almost no spatial pruning can be performed, slowing down the algorithm by several orders of magnitude. It is also worth noting, that in our setting, the algorithm performs worse on gaussian distributed data than on uniformly distributed data. This is explained by the fact that the space covered by a normal distribution with standard deviation x in each dimension, is generally larger than a hyper-rectangle with a side length of x in each dimension. A comparison of the runtime of **YLKS** and **PSR** w.r.t. the average AOL size is depicted in Figure 7.8(b) for both the uniform and the normal distributed datasets. The degree of uncertain has a similar influence on both **YLKS**and **PSR**.

7.6.5 Summary

The experiments presented in this section show that the theoretical analysis of our approach given in Section 7.5 can be confirmed empirically on both artificial and real-world data. The performance studies showed that our framework computing the rank probabilities indeed reduces the quadratic runtime complexity of state-of-the-art approaches to linear. Note that the cost required to pre-sort the object instances are neglected in our settings. It could be shown that our approach scales very well even for large databases. The speed-up gain of our approach w.r.t. the rank depth k has shown to be constant, which proofs that both approaches scale linear in k. Furthermore, we could observe that our approach is applicable for databases with a high degree of uncertainty (i.e. the degree of variance of the instance distribution).

7.7 Conclusions

In this chapter, we proposed a framework for efficient computation of probabilistic similarity ranking queries in uncertain vector databases. We introduced a novel concept that achieves a log-linear runtime complexity in contrast to the best-known existing approach that solve the same problem with quadratic runtime complexity. Our concepts are theoretically and empirically proved to be superior to all existing approaches. In an experimental evaluation, we showed that our approach scales very well and, thus, is applicable even for large databases. As future work, we plan to extend the concepts proposed in this chapter to further uncertainty models.

Chapter 8

Probabilistic Reverse k-Nearest Neighbor Queries on Uncertain Data

According to Definition 6, a Reverse k-Nearest Neighbor (RkNN) query retrieves all objects having a given query object as one of their k nearest neighbors. This chapter considers probabilistic reverse nearest neighbor (PRNN) queries, which return the uncertain objects having the query object as nearest neighbor with a sufficiently high probability. An algorithm is proposed to efficiently answering PRNN queries using new pruning mechanisms taking distance dependencies into account. Our experimental evaluation shows that our approach is able to significantly outperform previous state-of-the.art approaches. In addition, it is shown how this approach can easily be extended to PRkNN (where $k > 1$) query processing for which there is currently no efficient solution.

8.1 Introduction

The problem of RkNN query processing has been studied extensively on certain data [104, 173, 180]. However, due to the immense number of applications dealing with uncertain data, novel solutions to cope with uncertain objects are required. This chapter studies the problem of probabilistic reverse k-nearest neighbor (PRkNN) search in uncertain databases. A PRkNN query returns the set of objects having a sufficiently high probability to be the reverse k-nearest neighbor of a query object. Note that the query object can be uncertain as well.

There is a wide field of applications for PRNN queries (k=1), e.g. decision support, marketing, location-based services among others [44, 130]. For instance, consider a movie recommendation system that reports a list of movies that are similar to other movies that a user likes. What a user likes is however a very subjective variable that depends on user-specific preferences which cannot be measured. Therefore, each movie record is assumed to be associated with a set of user reviews, each consisting of a set of attribute values. Examples of such attributes are classification of the genre, humoristic value and suspense. An example for two such records is given in Figure 8.1. Thus, each movie record is represented by multiple records from different users. Differences between records of

UserID	Movie	Humor	Action
134	The Life of Brian	93 %	5 %
…	…	…	…
256	The Life of Brian	12 %	10 %
…	…	…	…

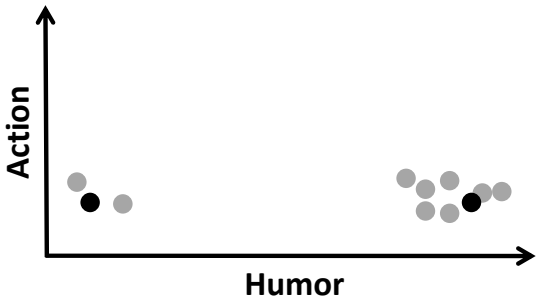
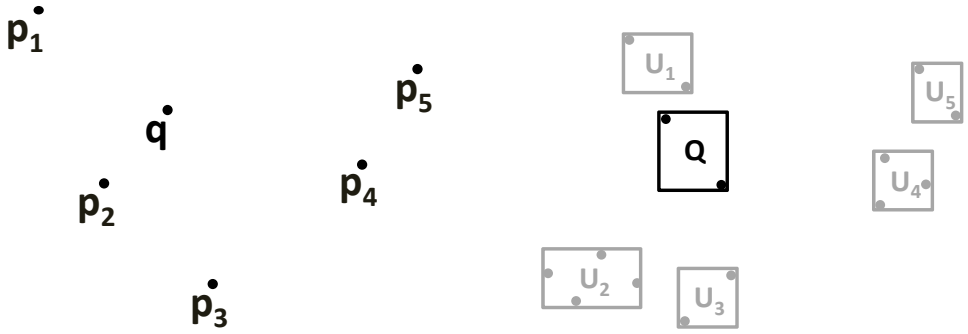


Figure 8.1: Uncertain object example: user ratings.

the same movie reflect the uncertainty in the user ratings. The advantage of using this uncertain data instead of simply using the average user recommendation of each movie record is shown by the following example: Consider a movie like "Monty Python's The Life of Brian" [43]: Many users will rate this movie as extremely funny. However, since this movie is based on a rather black sense of humor, some users may rate this movie as absolutely not funny. Thus, this movie would have an average of "moderately funny" and would result in a very large distance to other funny movies. Thus, this movie would never be recommended to users purchasing funny movies, even though there is a high probability that a user looking for funny movies may indeed be interested in this movie. Instead, the idea of this section is to consider such in a frequentistic way to estimate a probability that a user will find The Life of Brian to be a funny movie.

In this chapter, we first propose novel efficient methods for the PRNN (PRkNN with $k = 1$) query that outperforms the latest state-of-the-art solutions and then show how this approach can be extended to efficiently answer PRkNN queries (for $k \geq 1$) as a first solution of this problem. The contributions of this chapter can be summarized as follows:

- We propose a general framework for PRNN query algorithms and show how the two state-of-the-art approaches fit into it.
- By means of the spatial pruning criterion proposed in Chapter 5, we derive an efficient probabilistic pruning filter criterion. This criterion is shown to be correct in accordance the *possible worlds* semantics as it treats inherent distance dependencies in a correct way.
- We show how the new techniques for spatial pruning, probabilistic pruning and verification are combined to obtain an efficient PRNN algorithm. Additionally we show how this algorithm can be extended in order to answer PRkNN queries, which is the first solution to this problem.
- For the case where the objects are given by a discrete uncertainty model, we will show that all proposed techniques are correct and the algorithm efficiently yields the exact result. In the continuous case, where verification cannot be performed efficiently, the algorithm is able to approximate the exact result as tight as needed.
- We experimentally show that our proposed algorithm performs better than the two existing solutions under various settings. To the best of our knowledge, this is the first comparison of the two existing PRNN solutions.



(a) RNN example. (b) PRNN example.

Figure 8.2: Examples for RNN and PRNN.

The rest of this chapter is organized as follows: First we formally define the problem of PRNN queries in Section 8.2. Existing work related to PRNN queries is reviewed in Section 8.3. In Section 8.4, we describe the framework for PRNN query processing. In Section 8.5, we introduce novel spatial and probabilistic pruning filter criteria on uncertain data. The details of our implementation are presented in Section 8.6. An extension to the case of continuous uncertain is given in Section 8.7. In Section 8.8 we show how to extend the PRNN framework and algorithm to efficiently answer PRkNN queries. All proposed techniques are experimentally evaluated in Section 8.9. Finally Section 8.10 concludes this chapter.

8.2 Problem Definition

In this section, we give a formal definition of the Probabilistic Reverse Nearest Neighbor (PRNN) problem in uncertain databases. Therefore, we briefly review conventional reverse nearest neighbor queries and the uncertainty model used in this work.

Reverse k-nearest neighbor queries on (certain) point data have been defined Section 1.1.3. For recapitulation, consider the example illustrated in Figure 8.2(a). This figure shows the point object set ($P = \{p_1, ..., p_5\}$) with q as query object. Here, the result of an RNN query would contain the objects p_1 and p_2 using Euclidean distance. This result is evident, since p_1 and p_2 have q as its nearest neighbor, while point p_3 has point p_2 as its nearest neighbor, and points p_4 and p_5 have each other as nearest neighbors.

8.2.1 Uncertainty Model

Again using the X-tuple model, we assume the following: a probabilistic database $\mathcal{DB}$ is given by a set of uncertain objects $\mathcal{DB} = \{U_1, \ldots, U_n\}$ with d uncertain attributes. An uncertain object U_i is represented by a set of d-dimensional points $u_1, \ldots, u_m$ reflecting all possible instances of U_i. Each instance u_j is assigned with a probability $P(u_j)$ denoting the probability that U_i appears at u_j, i.e. all instances of U_i reflect the probability distribution of U_i. The probability distributions of each two objects are pairwise independent and the events of occurrence of all instances $u \in U_i$ are mutually exclusive. For clarity we

assume $P(U_i) = \Sigma_{j=1}^{m} u_j = 1$, although the proposed algorithms can easily be adapted to the case where $P(U_i) \leq 1$. A possible world $W = u_1, \ldots, u_n$ is a set of instances containing one instance from each object and occurring with an appearance probability of $P(W) = \Pi_{i=1}^{n} P(u_i)$. Let Ω denote the set of all possible worlds, then $\Sigma_{W \in \Omega} P(W) = 1$.

8.2.2 PRNN Queries in Uncertain Databases

For PRNN queries on uncertain databases the threshold parameter τ is introduced (cf. [44, 130]). Using τ as threshold, the user can restrict the result set to objects which have at least a predefined probability to be in the result in order to avoid reporting unnecessarily many results that are unlikely. A probabilistic nearest neighbor query $PRNN_Q^\tau$ then returns the set of all objects $U_i \in \mathcal{DB}$ where $P(U_i \in RNN_Q)$ (in the following denoted by $P(RNN_Q(U_i))) \geq \tau$. Naively, this probability can be calculated by performing a (non-probabilistic) RNN query on each possible world:

$$P(RNN_Q(U_i)) = \sum_{W \in \Omega} P(W) \cdot \mathcal{I}(U_i \in RNN(Q, W))$$

where $\mathcal{I}(U_i \in RNN(Q, W))$ is an indicator function that is 1 if U_i is a reverse nearest neighbor of Q in world W and 0 otherwise. In the example given in Figure 8.2(b), $RNN_Q(U_1)$ holds in all possible worlds, therefore $P(RNN_Q(U_1)) = 1$. In contrast, $P(RNN_Q(U_2)) = P(RNN_Q(U_4)) = P(RNN_Q(U_5)) = 0$, since there is no possible world in which $RNN_Q(U_2)$, $RNN_Q(U_4)$ or $RNN_Q(U_5)$ hold, as in each possible world the nearest neighbor of U_2 is U_1 and the nearest neighbor of U_4 is U_5 and vice versa. For U_3, we obtain the probability $P(RNN_Q(U_3))$ by building the sum of the probabilities of all possible worlds where at least one of the objects U_i ($i \in \{1, 2, 4, 5\}$) is closer to U_3 than Q to U_3. Obviously, this brute-force approach taking each possible world into account is in general not applicable because the number of possible worlds grows exponentially with the number of involved uncertain objects.

8.2.3 RNN Pruning

In order to shrink down the computational overhead of such queries, in this chapter we introduce efficient filter methods used to exclude (prune) as many objects as possible from the expensive query evaluation process. An object B can be pruned if we find another object A that is closer to B than the query object Q, i.e., if we find an object A that spatially dominates (c.f. Chapter 5) Q with respect to B. Following the notation of Chapter 6, we let $(A \prec_B Q)$ denote the random indicator variable that returns one if A dominates Q with respect to B and zero otherwise. In the context of RkNN queries, we say that A prunes B with respect to Q. The probability $P(A \prec_B Q)$ denotes the probability that A prunes B with respect to Q.

Naively, we can compute $P(A \prec_B Q)$ by simply adding the probabilities of all possible worlds in which A prunes B, exploiting inter-object independency:

$$P(A \prec_B Q) = \sum_{a_i \in A} \sum_{b_j \in B} \sum_{q_k \in Q} \mathcal{I}(dist(a_i, b_j) < dist(q_k, b_j)) \cdot P(a_i) \cdot P(b_j) \cdot P(q_k) \quad (8.1)$$

where $\mathcal{I}(dist(a_i, b_j) < dist(q_k, b_j))$ is an indicator function that returns 1 if $(dist(a_i, b_j) < dist(q_k, b_j))$ and 0 otherwise.

The problem of this naive approach is the computational cost of the triple-sum which is cubic in the number of instances of the uncertain objects. The number of instances x_i of an uncertain object X may in general be large, for example if the instances are obtained by Monte-Carlo sampling of an unknown PDF. Another problem is that the probability $P(A \prec_B Q)$ cannot directly be used to derive the probability that an object B is the RNN of Q. For two uncertain objects A_1 and A_2, the two events $A_1 \prec_B Q$ and $A_2 \prec_B Q$ are mutually dependent because both events depend on the assumptions made for object B (more details will be found in Section 8.5.3). In order to keep correctness w.r.t. the possible worlds semantics, this problem has to be taken into account.

8.3 Related Work

Reverse (k)-Nearest Neighbor (R(k)NN) queries on certain data have been studied for a long time [104, 173, 180, 185, 204, 206] including some work that I have been directly involved in ([108, 64, 5, 107, 63, 62]). Current state-of-the-art solutions use a filter-refinement approach to minimize the number of page accesses performed on the index organizing the data. The authors of [180], for example, perform an incremental nearest neighbor query in a best-first search manner where objects are organized in a spatial index and accessed with ascending distance to the query. Each accessed object is then used to prune other objects or index entries in a filter step. Finally, each remaining candidate has to be evaluated by means of a kNN query in a refinement step. Recently, we were able to improve the efficiency of existing RkkNN approaches in [108] by utilizing an aggregate R-Tree (aR-Tree [118, 144]). Furthermore, we proposed an efficient approach to return reverse nearest neighbors incrementally ([107]), i.e., returning iteratively the k'th nearest neighbor for $k = 1, ..., |\mathcal{DB}|$. An general approach for answering RkNN queries in general metric (non-necessarily Euclidean) spaces where the database is updated frequently, was developed by us in [5]. Details of the later publications are omitted in this thesis, as these early approaches do not consider uncertainty.

Yet, uncertainty in databases is a relatively new field and has received a lot of attention in the past few years. For Probabilistic Reverse Nearest Neighbor (PRNN) queries particularly two challenges arise: minimizing I/O-cost and minimizing CPU-cost. To the best of our knowledge, there are currently two approaches for answering PRNN queries. The approach from Chen et al. [130] which is designed for PRNN queries on uncertain objects represented by continuous probability density functions (PDFs) and the approach from

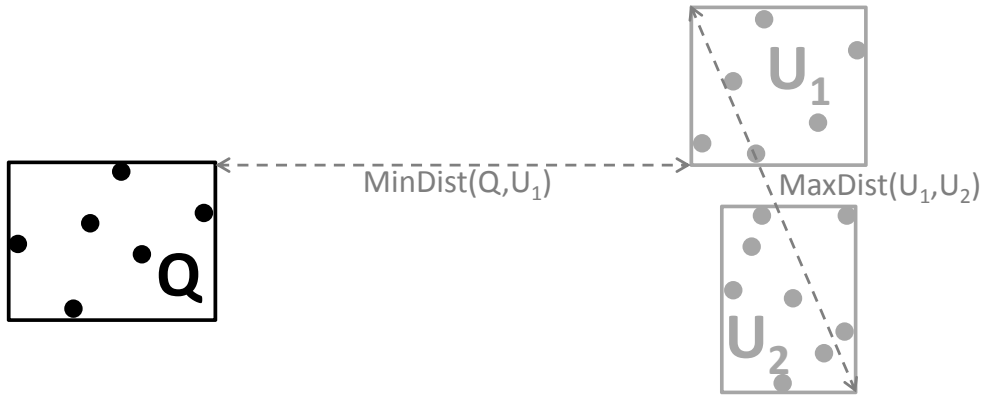


Figure 8.3: Pruning uncertain objects using minimal and maximal distance.

Cheema et al. [44] which works for the discrete case only. Both algorithms are discussed in more detail the next section.

8.4 PRNN Algorithm Sketch

Before introducing details of our PRNN query method, we will give a general framework for efficient PRNN query processing in an abstract fashion. For a comparison with state-of-the-art solutions, in Sections 8.4.5 and 8.4.6 we show how the two existing solutions by Cheema et al. [44] (in the following called CLWZP) and by Lian et al. [130] (in the following called LC) are implemented according to this framework.

8.4.1 Approximation of Objects

The probability distribution (or more specifically the uncertainty region) assigned to an uncertain object can become arbitrarily complex causing expensive distance computations at query time. A common solution to overcome this problem is to use conservative approximations, like spheres or rectangles providing efficient distance computation in a filter step. For efficient processing, these approximations are often organized in a hierarchical spatial index structure like the R-tree [82].

Consider Figure 8.3 for an example, where each uncertain object is represented by a minimum bounding rectangle (MBR) containing all possible instances of the object.

8.4.2 Spatial Pruning

It is possible to (spatially) prune objects without considering their probability distributions when using only the (spatial) approximations of the objects. Therefore, a pruning technique is needed. For instance an object B can be pruned by an object A for a query Q if $MaxDist(A, B) < MinDist(B, Q)$.

Example 20. *Consider again the example shown in Figure 8.3. For a R1NN query, object U_1 can be excluded from further consideration, as the maximal distance between U_1 and U_2 is smaller than the minimal distance between Q and U_1. Thus U_1 can never be R1NN of Q.*

The used pruning technique for uncertain objects efficiently organized by an index is easily extendable for pruning higher-level pages of the index. For example assume object U_1 in Figure 8.3 to be an index page containing several uncertain objects. Then all objects in this page can be pruned immediately.

8.4.3 Probabilistic Pruning

Probabilistic pruning is performed for objects that cannot be pruned spatially. In the probabilistic pruning step, the uncertainty regions of objects are partitioned. The aim of this partitioning is to prune more objects based on the probability threshold τ.

8.4.4 Verification

An object U_i which cannot be pruned by the pruning techniques is denoted as candidate. The next step requires each candidate to be verified, which means it has to be checked if $P(RNN_Q(U_i)) \geq \tau$. This involves finding all objects which affect this probability and considering these objects in more detail. The verification step is very expensive, since many possibilities have to be considered.

For a comparison with state-of-the-art solutions the following subsection show how the two existing solutions by Lian et al. [130] (in the following called LC) and by Cheema et al. [44] (in the following called CLWZP) are implemented according to this framework.

8.4.5 Framework Implementation: LC Algorithm

Approximation: This algorithm is designed for the case where the appearance probability of uncertain objects is represented as a continuous PDF. Though it can easily be adapted to the discrete case. Each uncertain object is approximated by a sphere.

Spatial Pruning: The proposed pruning technique is based on trigonometric functions and can only be applied for spherical objects. Thus, it cannot be directly applied to the index pages (the authors use an R-tree as index structure). To overcome this shortcoming, each (rectangular) page of the index is at runtime approximated by a sphere containing this page.

Probabilistic Pruning: Additionally, a second sphere is computed for each database object in a preprocessing step. This sphere has the same center as the first sphere, but the radius is chosen as the minimal radius covering instances with a cumulated probability of at least $1 - \tau$. The idea of this approach is that if this second sphere can be pruned, then the corresponding object is pruned with a probability of at least $1 - \tau$, so it must have a probability less than τ to be an RNN of Q, and thus, it cannot be a PRNN of Q.

Verification: In the verification step, a range query around each candidate U_i is issued. The result contains all objects U_j such that $MinDist(U_j, U_i) < MaxDist(U_i, Q)$, i.e. all objects which affect $P(RNN_Q(U_i))$. Then $P(RNN_Q(U_i))$ is calculated by considering all possible worlds of the involved objects.

8.4.6 Framework Implementation: CLWZP Algorithm

Approximation: The CLWZP algorithm uses minimum bounding rectangles for the approximation of the uncertain objects. Additionally, each uncertain object has a local R-tree which organizes its instances.

Spatial Pruning: The pruning is performed using several pruning techniques arranged in series. The first used technique is MinMax. As shown in Chapter 5, MinMax is not sufficient, which means that, based on rectangular approximations, MinMax cannot detect valid pruning in all cases. Therefore, a second technique is proposed for special spatial relations of the query object and the pruner. If this technique cannot be applied, a general technique is used which considers all corners of the pruner for prune evaluation (see [44] for details). All proposed techniques (except MinMax) generate a pruning region defined by the pruner and the query. In this region objects can safely be pruned.

Probabilistic Pruning: Probabilistic pruning utilizes the generated pruning regions. Based on these regions, it may happen that only parts of a prunee get pruned. In this case, the prunee is trimmed down and further represented by an MBR containing all instances which could not be pruned (using a computational geometry algorithm). Additionally, the authors propose to partition object Q, to further improve the pruning.

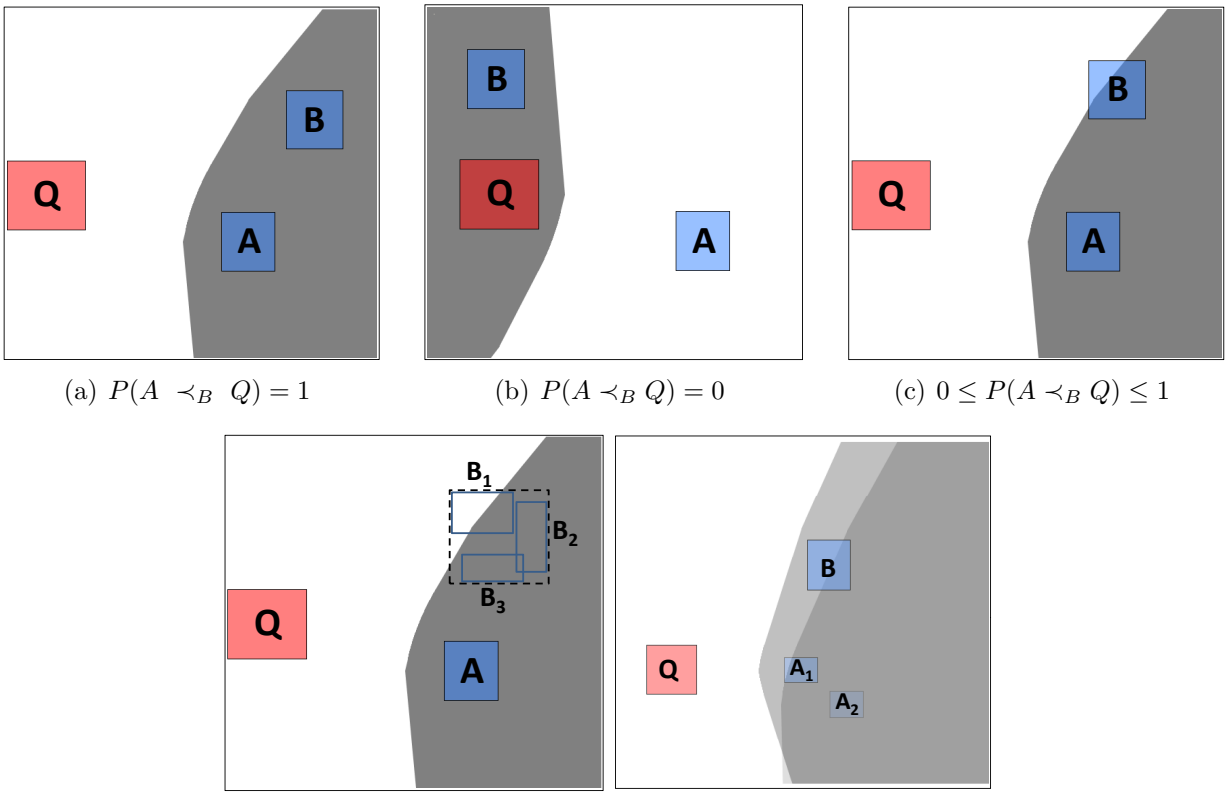
Verification: In the verification phase, a range query is issued for each candidate U_i containing all objects affecting $P(RNN_Q(U_i))$. For each instance u_i of a candidate, the instances of these objects are sorted by the distance to u_i and inserted in a list. Based on these lists it is possible to calculate $P(RNN_Q(U_i))$.

8.4.7 Discussion

Although the LC algorithm is the only PRNN algorithm so far which can handle uncertain objects represented by a continuous PDF, it has the following drawbacks:

Parameter τ: Since the probabilistic pruning sphere has to be pre-computed using τ, it is not possible to change τ at query time. In a dynamic query environment however, the parameter may be adapted to the user's preferences, which is not possible in this approach.

Spherical Approximation: The main challenge, especially for the higher-dimensional case, is to find a small enclosing sphere of an uncertain object for effective pruning results. Finding the smallest enclosing sphere of an arbitrarily shaped object however has exponential runtime (w.r.t. to the number of vertices of the object), which allows only finding good but not best possible spheres in reasonable computational time. Additionally, as stated in [130], the spatial pruning technique is only conservative but not optimal for dimensions larger than 2. A third problem regarding the spherical approximation is the approximation



(a) $P(A \prec_B Q) = 1$ (b) $P(A \prec_B Q) = 0$ (c) $0 \leq P(A \prec_B Q) \leq 1$

(d) Decomposition of B (e) Decomposition of A

Figure 8.4: Visualization of different pruning techniques (a)-(c) and object decomposition-ing (d)-(e).

of pages of the R-tree, which are rectangular by definition. A spherical approximation of an index page will therefore rarely be tight yielding low pruning power.

Verification: The verification step using integration of all remaining objects is based on the used uncertainty model. However, if the objects consist of discrete instances, there are more efficient solutions for the verification step (e.g. [125]). Note that also for the case where objects are represented by continuous PDFs, this step can be performed more efficiently, as we will show later.

The CLWZP algorithm on the other hand has a very complex spatial pruning technique which requires 2^d distance calculations in the worst case (where d is the dimensionality of the data). This makes the approach practically inapplicable for the high-dimensional case. Just as the LC pruning, the CLWZP pruning is conservative which means that there are cases where pruning is not performed although possible (we omit the proof due to space limitations, but will show this by experimental evidence). Regarding the probabilistic pruning of CLWZP, the problem is that trimming requires expensive geometric computation but is used extensively in the algorithm.

8.5 Hierarchical PRNN Processing

In this section, we propose our approach for PRNN processing implementing the above framework in consideration of the discrete uncertainty model. The complete algorithm of our PRNN query processing approach can be found in Section 8.6. In Section 8.7 we show how the proposed approaches can be extended to continuous uncertainty models.

8.5.1 Approximation

Similar to the approximation technique used for the CLWZP algorithm ([44]), in order to approximate uncertain objects we use minimum bounding boxes covering the uncertainty regions. This approximation is mainly used for spatial pruning. For probabilistic pruning, we need a more detailed object approximation. Therefore, we additionally assume that each uncertainty region of an object $X \in \mathcal{DB} \cup \{Q\}$ is hierarchically decomposed using a hierarchical space partitioning scheme. Specifically, we use an R*-tree [15] to hierarchically organize the instances of X. In addition to the spatial keys, each index entry stores the aggregated probability of all instances in the corresponding subtree. When traversing the index assigned to an uncertain object X in a breadth-first manner, each level of the R*-tree provides a disjoint and complete partitioning $\underline{\mathcal{X}}$ of X such that each partition $X' \in \underline{\mathcal{X}}$ contains a non-empty set of $m' \leq m$ instances $\{x'_1, ..., x'_{m'}\}$ and

$$\bigcup_{X' \in \underline{\mathcal{X}}} = \{x_1, ..., x_m\} \text{ and } \forall X'_i \in \underline{\mathcal{X}}, X'_{j \neq i} \in \underline{\mathcal{X}} : X'_i \cap X'_j = \emptyset.$$

An index entry representing partition X'_i contains the aggregated probability $P(X'_i) = \sum_{x'_j \in X'_i} P(x'_j)$. Note that disjoint property implies that any instance $x \in X$ is contained in at most one partition. The rectangular approximations of the instances contained in each R*-tree node however may overlap.

Let us note that we have to carefully select the refinement resolution since the PRNN computation is CPU-bound, as we will see in our experiments (cf. Section 8.9). We obtain a good control of this variable by using a very low R*-tree node capacity, e.g. in our experiments we used less than four entries per node.

To summarize, we use a memory-resident R*-tree to organize the objects $X \in \mathcal{DB}$ (global R*-Tree). The leaf entries containing the MBRs of the objects point to the local R*-trees of the objects. The local R*-trees are stored in a breadth-first manner, such that the highest levels of the trees can be obtained with one single scan.

8.5.2 Spatial Pruning

In accordance with our framework, here we propose a spatial pruning method that uses only the spatial keys of the uncertainty regions of the uncertain objects without taking into account further knowledge about the probability distribution. The following spatial pruning approach adopts the spatial domination concepts proposed in Chapter 5 in the

context of reverse k-nearest neighbor queries on certain data. Here, the main idea, is that if an object is pruned spatially, then its probability of being dominated equals one:

$$A^{\square} \prec_{B^{\square}} Q^{\square} \Rightarrow P(A \prec_B Q) = 1, \tag{8.2}$$

where $A^{\square}$, $B^{\square}$ and $Q^{\square}$ denote the minimum bounding rectangles of uncertain objects A, B and Q, respectively. These minimum bounding boxes are given by the root node of the R*-tree of each object. An example is given in Figure 8.4(a), where the uncertainty regions of the uncertain objects A, B and Q are depicted. In addition, the pruning region of A is shown in grey, that is the region containing exactly the points p in space for which it holds that p is definitely closer to all points in A than to any point in Q. Note that the pruning regions do not have to be materialized, here we only use them for illustration purposes. In this example, object B is completely contained in the pruning region of A and thus the predicate $A \prec_B Q$ holds. Exploiting Equation 6.1, we can safely prune B. In fact, to decide whether B is completely contained in the pruning region we only have to evaluate the predicate $DDC_{Optimal}(A, Q, B)$ defined in Definition 26 in Chapter 5.

In addition, we can use Equation 8.2 to find objects A that cannot prune B in any possible world, since the following equation holds:

$$Q^{\square} \prec_{B^{\square}} A^{\square} \Rightarrow P(A \prec_B Q) = 0 \tag{8.3}$$

This equation is evident, since in any world with instances $a_i \in A$, $b_j \in B$ and $q_k \in Q$, it holds that $dist(q_k, b_j) < dist(a_i, b_j) \Leftrightarrow \neg(dist(a_i, b_j) < dist(q_k, b_j))$ due to the anti-symmetry of $<$.

This equation allows to efficiently determine if object A cannot possibly prune B, by again evaluating $DDC_{Optimal}(Q, A, B)$. An example is given in Figure 8.4(b) where object B is completely contained in the pruning region of Q and thus, A cannot possibly prune B. Consequently, we can return any object $B \in \mathcal{DB}$ as a true hit, if for all objects $A \in \mathcal{DB} \setminus \{B\}$ the predicate $Q \prec_B A$ holds.

8.5.3 Probabilistic Pruning

Using the techniques proposed in the previous section, we can efficiently prune any uncertain object $B \in \mathcal{DB}$ for which it holds that there exists an uncertain object $A \in \mathcal{DB} \setminus \{B\}$ such that $A \prec_B Q$ holds, exploiting the predicate $DDC_{Optimal}(A, Q, B)$ defined in Definition 26 in Chapter 5. Now, we consider the case where the probability that A prunes B is between 0 and 1. In consideration of the possible worlds semantics, that means that there exist worlds in which A prunes B, but not all possible worlds satisfy this criterion. An exemplary situation is given in Figure 8.4(c): here, the uncertainty region of B is not completely contained in the pruning region of A. Thus, there may exist instances $b_i \in B$ that are not located in the pruning region of A. That in turn means, that there may be instances $a_i \in A$, $b_j \in B$ and $q_k \in Q$ such that $dist(b_j, q_k) < dist(b_j, a_i)$ and, thus, $P(A \prec_B Q) < 1$.

Individual Object Pruning

In general, the exact computation of $P(A \prec_B Q)$ is very expensive ($O(m^3)$). In this section we show how to compute lower/upper bounds of $P(A \prec_B Q)$ efficiently. This allows us to quickly detect in a filter step whether the probability that B can be pruned is at least $1-\tau$ (or cannot exceed τ) in order to prune B (or to return B as a true hit).

The key idea is to decompose the uncertainty region of an object X into subregions for which we know the probability that X is located in that subregion (cf. Section 8.5.1). Therefore, if $P(A \prec_B Q) < 1$, then there may still exist subregions $A' \subset A$, $B' \subset B$ and $Q' \subset Q$ such that $P(A' \prec_{B'} Q') = 1$. Examples of such situations are given in Figures 8.4(d) and 8.4(e). In Figure 8.4(d), some partitions of the uncertain object B are pruned by the uncertainty region of A. Given the total sum S of the probabilities of all partitions of B that cannot be pruned, we can conclude that B is an RNN of Q with a probability of at most S. In Figure 8.4(e), object A is divided into two partitions A_1 and A_2. It can be observed that B is fully contained in the pruning region of A_1 but not in the pruning region of A_2. Given the probability P of A_1, we can conclude that A prunes B with a probability of at least P. Thus, given a complete and disjoint object partitioning $\underline{\mathcal{A}}$, $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ as described in Section 8.5.1, we can identify triples of subregions ($A' \in \underline{\mathcal{A}}$, $B' \in \underline{\mathcal{B}}$, $Q' \in \underline{\mathcal{Q}}$) for which $P(A' \prec_{B'} Q') = 1$ (cf. Equation 6.1) holds.

Lemma 26. *Let A, B and Q be uncertain objects with disjoint and complete object decompositions $\underline{\mathcal{A}}, \underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$, respectively. To derive a correct lower bound $P_{LB}(A \prec_B Q)$ of the probability $P(A \prec_B Q)$ that A prunes B, we can accumulate the probabilities of combinations of these subregions as follows:*

$$P_{LB}(A \prec_B Q) =$$

$$\sum_{A' \in \underline{\mathcal{A}}, B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P(A') \cdot P(B') \cdot P(Q') \cdot (A' \prec_{B'} Q').$$

Proof. The probability of a combination (A', B', Q') can be computed by $P(A') \cdot P(B') \cdot P(Q')$ due to the assumption of mutually independent objects. These probabilities can be aggregated due to the assumption of disjoint subregions, which implies that any two different combinations of subregions ($A' \in \underline{\mathcal{A}}$, $B' \in \underline{\mathcal{B}}$, $Q' \in \underline{\mathcal{Q}}$) and ($A'' \in \underline{\mathcal{A}}$, $B'' \in \underline{\mathcal{B}}$, $Q'' \in \underline{\mathcal{Q}}$, $A' \neq A'' \wedge B' \neq B'' \wedge Q' \neq Q''$) must represent disjoint sets of possible worlds. By definition it holds that, if $(A' \prec_{B'} Q') = 1$, then A prunes B in all possible worlds defined by combinations of instances ($a_i \in A', b_j \in B', q_k \in Q'$). But not all possible worlds where A prunes B are covered by these combinations and, thus, do not contribute to $P_{LB}(A \prec_B Q)$. Consequently, $P_{LB}(A \prec_B Q)$ lower bounds $P(A \prec_B Q)$. □

Analogously, we can define an upper bound of $P(A \prec_B Q)$ using the intuition of Equation 8.3:

Lemma 27. *An upper bound $P_{UB}(A \prec_B Q)$ of $P(A \prec_B Q)$ can be derived as follows:*

$$P_{UB}(A \prec_B Q) = 1 - P_{LB}(Q \prec_B A).$$

Proof. This equation is evident due to Lemma 17 and the observation that in any world where $A = a$, $B = b$ and $R = r$ it holds that $(a \prec_r b) = 1 \Leftrightarrow (b \prec_r a) = 0$. □

Following the partitioning concept proposed in Section 8.5.1, we can control the resolution and can refine the partitioning on demand. Naturally, the more refined the object partitions are, the tighter are the bounds that can be computed but the higher are the corresponding cost of deriving them. In particular, starting from the entire MBRs of the objects, we can progressively partition them by traversing the object instance index to iteratively derive tighter probability bounds until a desired degree of certainty is achieved (based on some threshold). In general, this allows us to significantly prune the space of candidate objects. However, the RNN probability of a given object B cannot be straightforwardly derived with the use of these bounds, as we will see in the following section.

Joint Object Pruning

Given the probability bounds $P_{LB}(A \prec_B Q)$ and $P_{UB}(A \prec_B Q)$, the next problem is to accumulate these probabilities to obtain an approximation of the probability $P(RNN_Q(B))$ that object B is an RNN of Q. The problem at issue is that, though all objects are assumed to be mutually independent, the two events $A_1 \prec_B Q$ and $A_2 \prec_B Q$ are generally mutually dependent as discussed in Section 6.4.1.

To avoid this problem, we present a way to conservatively approximate the probability $P(RNN_Q(B))$ while accounting for the dependencies between the random variables $A_i \prec_B Q$ ($A_i \in \mathcal{DB}$).

Decomposition of Database Object A_i

Let $I = \{A_1, ..., A_{|I|}\}$ denote the set of *influence objects* of B, which neither completely prune B w.r.t. Q nor are completely pruned by B. We only have to consider the set of random events $\{A_1 \prec_B Q, ..., A_{|I|} \prec_B Q\}$, since the objects A_i for which $P(A_i \prec_B Q) = 0$ holds do not have any influence on $P(RNN_Q(B))$, and if there exists an object A_i for which it holds that $P(A_i \prec_B Q) = 1$, then we can already conclude that $P(RNN_Q(B)) = 0$.

Due to the problem of mutual dependencies between pruning events we cannot simply use the probability bounds $P_{LB}(A_i \prec_B Q)$ and $P_{UB}(A_i \prec_B Q)$ directly, as this would yield incorrect results. However, we can use the observation that the objects A_i are mutually independent and each candidate object A_i only appears in a single random variable $A_1 \prec_B Q, ..., A_{|I|} \prec_B Q$. Exploiting this observation, we can decompose, e.g. using the bounding boxes of the R^*-trees of the objects, the objects $A_1, ..., A_{|I|}$ to obtain mutually independent bounds for the probabilities $P(A_1 \prec_B Q), ..., P(A_{|I|} \prec_B Q)$, as stated by the following lemma:

Lemma 28. *If B and Q are not decomposed, i.e. if $\underline{\mathcal{B}} = \{B\}$ and $\underline{\mathcal{Q}} = \{Q\}$, then $P(RNN_Q(B))$ is lower bounded by*

$$P_{LB}(RNN_Q(B)) = \prod_{A_i \in I} 1 - P_{UB}(A_i \prec_B Q).$$

Proof. The event that B is an RNN of Q is (by definition) equal to the event that no database object A_i in I prunes B. Due to Corollary 8 (see below), the event $A_i \prec_B Q$ is independent of the computation of $P_{LB}(A_j \prec_B Q)$ for $i \neq j$. Therefore, the bounds $P_{LB}(A_i \prec_B Q)$ are also independent of $P_{LB}(A_j \prec_B Q)$. This independence allows to derive a lower bound of the joint probability that all objects $A_1, ..., A_{|I|}$ prune B by simply taking the product of the bounds $\prod_{i=1}^{|I|} P_{LB}(A_i \prec_B Q)$. The same applies for the joint probability of the complementary events (i.e. the probability of the events that the A_i's do not prune B). Since we only have bounds of $P(A_i \prec_B Q)$, we need to use the upper bounds $P_{UB}(A_i \prec_B Q)$ in order to minimize the complementary probability $1 - P_{UB}(A_i \prec_B Q)$ to derive a lower bound of the event that no A_i prunes B. □

An upper bound can be derived analogously:

$$P_{UB}(RNN_Q(B)) = \prod_{A_i \in I} 1 - P_{LB}(A_i \prec_B Q).$$

In summary, we can now derive, for each uncertain candidate object B a lower and an upper bound of the probability that B is an RNN of Q. However, these bounds may still be rather loose, since we only consider the full uncertainty region of B and Q so far, without any decomposition. In the following section, we will show how to obtain more accurate, still mutually independent probability bounds based on decompositions of B and Q.

Decomposition of Candidate and Query Object

Since the uncertain objects B and Q appear in each random event $A_i \prec_B Q$ ($A_i \in \mathcal{DB}$) that has to be evaluated, we cannot split the objects B and Q independently. Intuitively, the reason for this dependency is that any knowledge about the random event $A_i \prec_B Q$ may impose constraints on the position of B and Q. However, Lemma 28 directly yields the following corollary:

Corollary 8. *Given partitions $B' \subseteq B$ and $Q' \subseteq Q$. Under the condition that the location of B is in B' and that the location of Q is in Q', we get*

$$P_{LB}(RNN_{Q'}(B')) = \prod_{A_i \in I} 1 - P_{UB}(A_i \prec_{B'} Q').$$

Note that the probability $P_{LB}(RNN_{Q'}(B'))$ is equal to the probability $P_{LB}(RNN_Q(B)|B \in B', Q \in Q')$, where $B \in B', Q \in Q'$ is a constraint to all possible worlds where B is located in partition B' and Q is located in partition Q'. This allows us to individually consider the subset of possible worlds where $B \in B'$ and $Q \in Q'$ and use Lemma 8 to efficiently compute $P_{LB}(RNN_{Q'}(B'))$ and $P_{UB}(RNN_{Q'}(B'))$. This can be performed for each pair $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$, where $\underline{\mathcal{B}}$ and $\underline{\mathcal{Q}}$ denote the decompositions of B and Q, respectively. Now, we can treat pairs of partitions $(B', Q') \in \underline{\mathcal{B}} \times \underline{\mathcal{Q}}$ independently, since all pairs of partitions represent disjoint sets of possible worlds due to the assumption of a disjoint partitioning. Exploiting this independency, we can derive tighter bounds $P_{LB}(RNN_Q(B))$

and $P_{UB}(RNN_Q(B))$ for the probability that B is an RNN of Q by computing a lower and an upper bound of $P(RNN_{Q'}(B'))$ for each $(Q' \in \underline{\mathcal{Q}})$ and each $(B' \in \underline{\mathcal{B}})$ and then computing the weighted sum of these bounds as follows:

$$P_{LB}(RNN_Q(B)) =$$

$$\sum_{B' \in \underline{\mathcal{B}}, Q' \in \underline{\mathcal{Q}}} P_{LB}(RNN_{Q'}(B')) \cdot P(B') \cdot P(Q'). \tag{8.4}$$

8.5.4 Verification

For the verification step, we perform, for each remaining candidate B, a probabilistic nearest neighbor query using the algorithm proposed in Chapter 7 for probabilistic ranking queries (and setting $k = 1$). This algorithm takes Q, B and $\mathcal{DB} \setminus B$ as input and returns $P(NN_B(Q))$ which is equivalent to $P(RNN_Q(B))$. If this value is above τ, then B is returned as result, otherwise it is discarded. This algorithm avoids enumeration of all (exponentially many) possible worlds by sorting the instances of the influence objects I w.r.t. the distance to B and performing a distance browsing on this sorted list.

8.5.5 Complexity Analysis

In this section we will analyze the runtime complexity of each part of the proposed PRNN algorithm:

Spatial Pruning: The basic spatial pruning takes each pair of objects $A, B \in \mathcal{DB}$ where $A \neq B$ and checks whether $A \prec_B Q$ holds. Thus the runtime is $O(|\mathcal{DB}|^2)$. In the average case, this step can be accelerated by the use of a spatial index structure to $(O(|\mathcal{DB}| \cdot log(|\mathcal{DB}|)))$, but the worst-case runtime remains $O(|\mathcal{DB}|^2)$. The spatial pruning provides us with a set of candidate objects S_{cnd} where each candidate $C_i \in S_{cnd}$ is associated with a set of influence objects S^i_{ifl}.

Probabilistic Pruning: The probabilistic pruning step considers each candidate object C_i separately and partitions the candidate object, the query Q and the influence objects S^i_{ifl}. Assuming a branching factor of the spatial index of b, each object consists of at most b^{depth} partitions, where $depth$ is a parameter chosen before query processing. The pruning relation $A' \prec_{C_i'} Q'$ is used for each pair of partitions $Q' \in Q, C_i' \in C_i$ and each partition A' of objects A in S^i_{ifl}. This leads to a runtime of $O(b^{2 \cdot depth} \cdot |S^i_{ifl}| \cdot b^{depth}) = O(|S^i_{ifl}| \cdot b^{3 \cdot depth})$ for each candidate C_i. In the worst case, where $|S^i_{ifl}| \in O(|\mathcal{DB}|)$, $|S_{cnd}| \in O(|\mathcal{DB}|)$ and $b^{depth} = m$ this yields a total runtime of $O(|\mathcal{DB}|^2 \cdot m^3)$, where m is the number of instances in each object.

Verification: After the probabilistic pruning step, a smaller set of candidates $S'_{cnd} (\subseteq S_{cnd})$ remains. For each candidate $C_i \in S'_{cnd}$ verification is performed. Using the algorithm proposed in Chapter 7, this requires to sort all $m \cdot |S^i_{ifl}|$ instances of objects in S^i_{ifl} according to all m instances in C_i. We derive a runtime of $O(|S'_{cnd}| \cdot |S^i_{ifl}| \cdot m^2 \cdot log(|S^i_{ifl}| \cdot m))$. In the worst case, this is in $O(|\mathcal{DB}|^2 \cdot log(|\mathcal{DB}|)$, assuming m to be constant.

The parameter *depth*: It can be observed, that for the case where m is large, the runtime of the probabilistic pruning step may exceed the runtime of the verification step. In our experimental section, we will verify this observation, and show how to choose values for the parameter *depth* such that this problem is avoided.

8.6 Implementation

8.6.1 Overview

Algorithm 5 combines the main modules for PRNN processing. The input requires an uncertain query object Q, an R-tree based index structure organizing the uncertain objects from the database I_{DB}, a probability threshold τ and a parameter *depth* controlling the depth of our probabilistic pruning computation. The spatialPruning method fills the sets S_{cnd} with potential result objects and S_{prn} with objects (entries) which can certainly be excluded using the spatial pruning technique proposed in Section 8.5.2. For each remaining object $B \in S_{cnd}$, the getInfluenceObjects method returns a set (S_{ifl}) of all objects from the two sets (S_{cnd} and S_{prn}) which could influence $P(RNN_Q(B))$. With these objects, probabilistic pruning according to Section 8.5.3 is performed. Depending on the result, the candidate is either discarded, added to the result set or verified. In the latter case, computation following [125] and [44] is performed to calculate the exact $P(RNN_Q(B))$. If this probability is above τ, the candidate can be confirmed as a result. In the following, we explain the individual modules in detail.

Algorithm 5 PRNN query processing

Require: Q, I_{DB}, τ, *depth*
1: $S_{cnd} = \emptyset$, $S_{prn} = \emptyset$
2: spatialPruning(Q, I_{DB}, S_{cnd}, S_{prn})
3:
4: $S_{res} = \emptyset$
5: **for** each $B \in S_{cnd}$ **do**
6: S_{ifl} = getInfluenceObjects(Q, B, $S_{cnd} \setminus \{B\}$, S_{prn})
7: i := probabilisticPruning(Q, B, S_{ifl}, τ, *depth*)
8: **if** i=1 **then**
9: *//B cannot be RNN of Q*
10: **else if** i=-1 **then**
11: *//B is RNN of Q*
12: $S_{res} = S_{res} \cup \{B\}$
13: **else**
14: *//B has to be verified*
15: **if** verify(Q, B, S_{ifl}, τ) **then**
16: $S_{res} = S_{res} \cup \{B\}$
17: **end if**
18: **end if**
19: **end for**
20: **return** S_{res}

8.6.2 Spatial Pruning

The spatialPruning method (cf. Algorithm 6) performs a best-first search using a heap H prioritized by $minDist(Q, e)$, where e is an entry of the index I_{DB}. The heap is implemented such that the set of contained objects can be accessed without destroying the heap structure. For each de-heaped entry e, the predicate $e_2 \prec_e Q$ checks if this entry is pruned by another object or entry e_2 (contained in H, S_{prn} or S_{cnd}) according to Q, using the pruning technique as described in Section 8.5.2. If it can be pruned, it is inserted in the S_{prn} set. Otherwise, if e contains a data object, it is added to the candidate set S_{cnd} and if e is a directory entry, its children are inserted into the heap.

Algorithm 6 spatialPruning

Require: Q, I_{DB}, S_{cnd}, S_{prn}
1: init min-heap H with root entry of I_{DB}
2: **while** H is not empty **do**
3: de-heap an entry e from H
4: **if** $\exists e_2 \in H \cup S_{prn} \cup S_{cnd} : e_2 \prec_e Q$ **then**
5: $S_{prn} = S_{prn} \cup \{e\}$
6: **else if** e is directory entry **then**
7: **for** each child ch in e **do**
8: insert ch in H
9: **end for**
10: **else if** e is data entry **then**
11: $S_{cnd} = S_{cnd} \cup \{e\}$
12: **end if**
13: **end while**

8.6.3 Obtaining Influence Objects

For each candidate B, it is important for the next steps to find the objects which influence $P(RNN_Q(B))$. This is done by Algorithm 7, which exploits Implication 8.3 in order to determine those objects that cannot possibly prune B. Therefore, each data entry e for which $Q \prec_B e$ does not hold is inserted into the S_{ifl} set.

8.6.4 Probabilistic Pruning

The goal of the probabilistic pruning is to estimate $P(RNN_Q(B))$ for a candidate B in a best possible way. If we can detect early that $P(RNN_Q(B)) > \tau$ or $P(RNN_Q(B)) < \tau$, further computation can be saved. The parameter *depth* is used to control how deep the hierarchical index structures (organizing the instances of each uncertain object) are resolved for more accurate pruning. Thus, the parameter offers to define a trade-off between accuracy and computational efficiency in the probabilistic pruning step. In each iteration the involved objects (Q, B and all $I \in S_{ifl}$) are partitioned, which means we consider the partitioning at the i^{th} level of each local R*-tree. According to Section 8.5.3, each combination of $(Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}})$ must be treated independently. Thus, for each of these

Algorithm 7 getInfluenceObjects

Require: Q, B, S_{cnd}, S_{prn}
1: $S_{ifl} = \emptyset$
2: **for** each $e \in S_{prn} \cup S_{cnd}$ **do**
3: **if** $\neg(Q \prec_B e)$ **then**
4: **if** e is directory entry **then**
5: $S_{prn} = S_{prn} \setminus \{e\}$
6: **for** each child ch in e **do**
7: $S_{prn} = S_{prn} \cup \{ch\}$
8: **end for**
9: **else if** e is data entry **then**
10: $S_{ifl} = S_{ifl} \cup \{e\}$
11: **end if**
12: **end if**
13: **end for**
14: **return** S_{ifl}

combinations, we first obtain bounds $P_{LB}(I \prec'_B Q')$ and $P_{UB}(I \prec'_B Q')$ of the probability that each $I \in S_{ifl}$ prunes B' w.r.t. Q', using Lemmas 26 and 27. Using Corollary 28, we multiply the complement of these bounds to acquire the lower (upper) bound probability $P_{LB}(RNN_{Q'}(B'))$ ($P_{UB}(RNN_{Q'}(B'))$) that no object $I \in S_{ifl}$ prunes B' according to Q'. Since all combinations ($Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}}$) are independent, the results can be summed up (weighting with the possible world probability of ($Q' \in \underline{\mathcal{Q}}, B' \in \underline{\mathcal{B}}$)), to obtain the global probability bounds $P_{LB}(RNN_Q(B))$ and $P_{UB}(RNN_Q(B))$ according to Equation 8.4. This method returns -1 if the candidate B is PRNN of Q (with τ as threshold), and 1 if B can be pruned. If *depth* is not set to the maximum height of the R*-trees, it is possible that no decision can be made. In this case the method returns 0.

8.7 Continuous Distributions

In this section, we show how our approach can be extended to continuously distributed uncertainty models. Again, we assume that the database $\mathcal{DB}$ consists of multi-attribute objects $o_1, ..., o_N$ that may have uncertain attribute values. An uncertain attribute is defined as follows:

Definition 33 (Continuous Probabilistic Attribute)**.** *A continuous probabilistic attribute* attr *of an object X is a random variable drawn from a probability distribution with density function f_i^{attr}.*

An uncertain object X has at least one uncertain attribute value. The function $f_X(x)$ denotes the multi-dimensional probability density function (PDF) of o_i that combines all density functions for all probabilistic attributes *attr* of X.

Approximation: Following the convention of continuous uncertain databases [31, 45, 48, 51, 55, 131, 170], we assume that each uncertain object X is (minimally) bounded by

Algorithm 8 probabilisticPruning

Require: Q, B, S_{ifl}, τ, *depth*
1: **for** $1 \ldots depth$ **do**
2: split(Q)
3: split(B)
4: $\forall I \in S_{ifl}$ split(I)
5: $P_{LB}(RNN_Q(B)) = 0, P_{UB}(RNN_Q(B)) = 0$
6: **for all** $Q' \in Q$ and $B' \in B$ **do**
7: $P_{LB}(RNN_{Q'}(B')) = 1, P_{UB}(RNN_{Q'}(B')) = 1$
8: **for** each $I \in S_{ifl}$ **do**
9: $P_{LB}(I \prec_{B'} Q') = 0$, $P_{UB}(I \prec_{B'} Q') = 1$
10: **for** each $I' \in I$ **do**
11: **if** $(I' \prec_{B'} Q')$ **then**
12: $P_{LB}(I \prec_{B'} Q') = P_{LB}(I \prec_{B'} Q') + P(I')$
13: **else if** $(Q' \prec_{B'} I')$ **then**
14: $P_{UB}(I \prec_{B'} Q') = P_{UB}(I \prec_{B'} Q') - P(I')$
15: **end if**
16: **end for**
17: $P_{UB}(RNN_{Q'}(B')) = P_{UB}(RNN_{Q'}(B')) \cdot (1.0 - P_{LB}(I \prec_{B'} Q'))$
18: $P_{LB}(RNN_{Q'}(B')) = P_{LB}(RNN_{Q'}(B')) \cdot (1.0 - P_{UB}(I \prec_{B'} Q'))$
19: **end for**
20: $P_{LB}(RNN_Q(B)) = P_{LB}(RNN_Q(B) + P(Q')) \cdot P(B') \cdot P_{LB}(RNN_{Q'}(B'))$
21: $P_{UB}(RNN_Q(B)) = P_{UB}(RNN_Q(B) + P(Q')) \cdot P(B') \cdot P_{UB}(RNN_{Q'}(B'))$
22: **end for**
23: **if** $P_{LB}(RNN_Q(B)) > \tau$ **then**
24: **return** -1
25: **else if** $P_{UB}(RNN_Q(B)) < \tau$ **then**
26: **return** 1
27: **end if**
28: **end for**
29: **return** 0

a rectangular *uncertainty region* $X^{\square}$ such that $\forall x \notin X^{\square} : f_X(x) = 0$ and

$$\int_{X^{\square}} f_X(x)dx \leq 1.$$

If f_i is an unbounded PDF, e.g., a Gaussian PDF, we truncate PDF tails with negligible probabilities and normalize the resulting PDF. This procedure is also used in related work [45, 48, 31]. Specifically, [31] shows that, for a reasonably low truncation threshold, the impact on the accuracy of probabilistic ranking queries is very low.

Spatial Pruning: Since our spatial pruning approach (cf. Section 8.5.2) is based on rectangular approximations only, it can be applied on continuously distributed objects without any adaptations.

Probabilistic Pruning: Our probabilistic pruning approach (cf. Section 8.5.3) applies disjoint and complete partitioning schemes $\underline{\mathcal{X}}$ $(X \in \mathcal{DB} \cup Q)$ to conservatively and progressively approximate the probability $P(RNN_Q(B))$ that an object B is an RNN of Q. This technique is also applicable for partitions $X' \in \underline{\mathcal{X}}$ for which the probabilities

$P(X')$ are known. To derive a complete and disjoint partitioning scheme on continuous uncertain objects, we propose to apply a kd-tree [17] having $X^{\square}$ in its root. In each level of the tree, each node is split with respect to its median in dimension d. The split-dimension d is rotated for each level of the tree. The advantage of this partitioning scheme is that the probability of a node on level i (here level 0 denotes the root level) of the tree, has a total probability of $1/2^i$. Due to the nature of continuous PDFs, this kd-tree has an infinite height. Therefore, we propose to restrict the height of the kd-tree (i.e. the maximum number of splits). The maximum number of splits is denoted by *depth*.

Verification: To compute the exact probability $P(A \prec_B Q)$ that an object A prunes B for a PRNN query with query object Q, we require to compute the following integral:

$$\int_{a \in A^{\square}} \int_{b \in B^{\square}} \int_{q \in Q^{\square}} f_A(a) \cdot f_B(b) \cdot f_Q(q) \cdot (a \prec_b q) da\, db\, dq.$$

This computation requires expensive numeric integrations, since in general the integral of the PDF f_X of an uncertain object may not be representable as a closed-form expression and the integral of $(a \prec_b q)$ does not have a closed-form expression. The computation of $RNN_Q(B)$ is even more expensive, since to compute $RNN_Q(B)$, the PDFs of all database objects may need to be considered to avoid dependencies. Therefore, we propose to avoid verification by using a large *depth* parameter, so that the derived probability bounds $P_{LB}(RNN_Q(B))$ and $P_{UB}(RNN_Q(B))$ become very tight and with a high probability, no more candidates remain after the probabilistic pruning step. For the remaining candidates, the best we can do is an efficient approximation ([124]). Therefore, we propose the following strategies:

- Our approach uses the derived probability bounds of candidates to bound the error. In many applications, this bound may be sufficient to the user and verification may be avoided.
- We can adapt the techniques as proposed by [124] to approximate the object PDFs using cubic splines that have a closed-form solution, and approximate the rank of Q w.r.t. B.

8.8 Probabilistic RkNN Queries

In this section we show how our proposed techniques can be extended to probabilistic RkNN queries. An RkNN query is defined as follows: Given a set of (certain) points P, a query object q, and a positive integer k, a reverse nearest neighbor query (RNN_q) returns all $p \in P$ which have q in their k-nearest neighbor set, formally ([180]): $RkNN_q = \{p \in P | dist(p,q) \leq dist(p,p_k)\}$, where p_k is the k-th nearest neighbor of p. In the context of uncertain objects, a $PRkNN_Q^{\tau}$ query returns the set of all objects $U_i \in \mathcal{DB}$, for which the probability $P(U_i \in RkNN_Q)$ that U_i is a RkNN of Q is at least τ.

Approximation: Analogous to the $k = 1$ case, we approximate uncertain objects using MBRs and hierarchical partitioning.

Spatial Pruning: In the case where $k > 1$, the random event $A_i \prec_B Q$ must hold for at least k uncertain objects $A_i, 1 \leq i \leq |\mathcal{DB}|$ in order to prune candidate B. Therefore, an uncertain candidate object can safely be pruned if there exist at least k objects A_i for which the complete pruning criterion $DDC_{Optimal}(A_i, Q, B)$ holds. The complexity for the spatial pruning step is $O(|\mathcal{DB}|^2)$ and independent of k, since in the worst case where a candidate B cannot be pruned, all objects may have to be considered.

Probabilistic Pruning: For probabilistic pruning, the task is to determine for a candidate object B, if its probability to be RkNN of Q is definitely less than τ (at least τ) in order to prune B (return B as a true hit). Analogous to the $k = 1$ case, we can derive the following bounds for the probability $P(A \prec_B Q)$ that $A \in \mathcal{DB} \setminus B$ is closer to B than Q: a lower bound $P_{LB}(A \prec_B Q)$ (using Lemma 26) and an upper bound $P_{UB}(A \prec_B Q)$ (using Lemma 27). Given these bounds, we can apply the concept of uncertain generating functions proposed in Section 6.4.3 in order to compute for each $0 \leq j < k$ a lower bound $P_{LB}(\#Pruners = j)$ and an upper bound $P_{UB}(\#Pruners = j)$ of the probability of the random event that for exactly j uncertain objects A_i, $A_i \prec_B Q$ is true. Bounds for the probability of the event $RkNN_Q(U_i)$ that U_i is a RkNN of Q can then be derived as follows:

$$P_{LB}(RkNN_Q(U_i)) = \sum_{0 \leq j < k} P_{LB}(\#Pruners = j)$$

$$P_{UB}(RkNN_Q(U_i)) = \sum_{0 \leq j < k} P_{UB}(\#Pruners = j)$$

An uncertain object U_i can be pruned if $P_{UB}(RkNN_Q(U_i)) < \tau$ and returned as a true hit if $P_{LB}(RkNN_Q(U_i)) > \tau$.

As shown in Section 6.4.3, the computational complexity is linear in k, yielding a total of $O(|DB|^2 \times k)$ for the probabilistic pruning.

Verification: The verification step can be performed analogously to the $k = 1$ case using the algorithm proposed in Chapter 7, which has been designed for $k \geq 1$. The total complexity of this algorithm is $O(|\mathcal{DB}|^2 \cdot log(|\mathcal{DB}|) + k \cdot |\mathcal{DB}|^2)$.

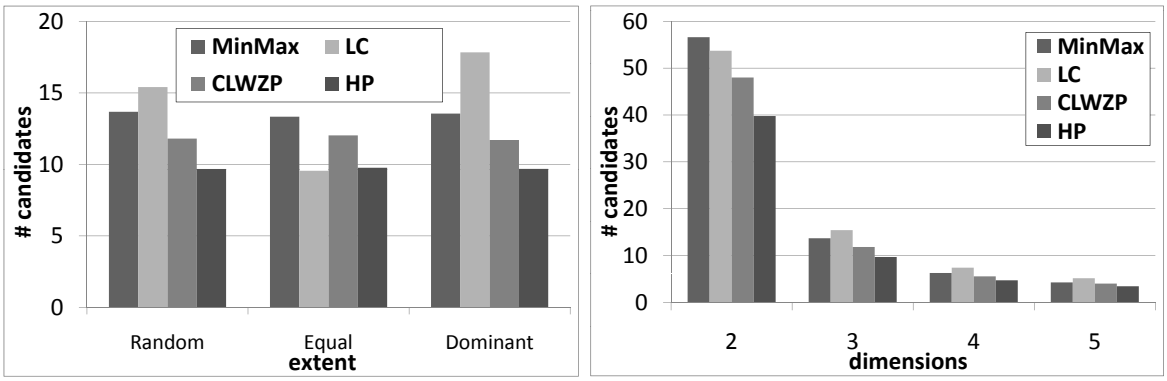
parameter	values synthetic	values real
db size	**2000** - 10000	6216
dimensionality	2, **3**, 4, 5	2
# instances	50, **100**, 200, 400	100
τ	0.1 **0.2**, 0.3	**0.2**
maxdepth	0, 1, **2**, 3, 4	**2**
MBR*extent*	0.01, 0.02, 0.03, 0.04, **0.05**	N/A

Table 8.1: Parameters and their default values.

8.9 Experiments

In the experimental section, we compare our approach which we call HP (hierarchical pruning) with the two state-of-the-art PRNN query algorithms CLWZP [44] and LC [130]. For the implementation of CLWZP and LC we replaced R-trees by R*-trees wherever they were used. For the global R^*-tree we use a page size of 1024 byte. For the local R^*-trees indexing the instances of an uncertain object, we set the page size to a maximal capacity of three entries. The page size was chosen small in order to minimize CPU-cost, which we will see is the main bottleneck of a PRNN query. We tested the three approaches under various parameter and data settings. For each setting, we performed 100 queries and averaged the measures. Since our experiments have been conducted against discrete uncertain datasets, we have adapted the LC algorithm to the discrete case for a fair comparison. The involved parameters and their default values (bold) can be seen in Table 8.1. For our experiments, we used one real-world dataset and several synthetic datasets to show the effect of changes in dimensionality and size. The datasets are described as follows: We generate synthetic uncertain objects in the $[0, 1]^d$ space by uniformly selecting the expected position of the objects in the space. A rectangle is generated around the expected position with a fixed total sum of side lengths (referred to as *extent*) with a default value of 0.05. By default, the extent is distributed uniformly on the dimensions, so there is a diversity of MBR shapes, some that are nearly cuboid, while others have a very large extent in few dimensions only. The object instances within the MBR are distributed uniformly by default. As a real-world dataset, we utilize the International Ice Patrol (IIP) Iceberg Sightings Dataset[1]. This data set contains information about iceberg activity in the North Atlantic in the years 1960 to 2010. It contains the latitude and longitude values of 6216 sighted icebergs. An uncertain object is generated for each iceberg, by generating 100 Normal-distributed instances having a mean value corresponding to the position of its most recent sighting at the date 31.12.2009 and a variance corresponding to the time period between this date and the sighting. To avoid extreme impact of icebergs that have not been seen for decades, any instances outside a square of extent 0.0004 centered at the mean are cut off. The experiments were run on a Windows 7 notebook with an Intel Core i5 processor (2.27 GHz), 6GB RAM.

[1]The IIP dataset is available at the National Snow and Ice Data Center (NSIDC) web site (*http://nsidc.org/data/g00807.html*).



(a) Varying extents.

(b) Varying dimensions.

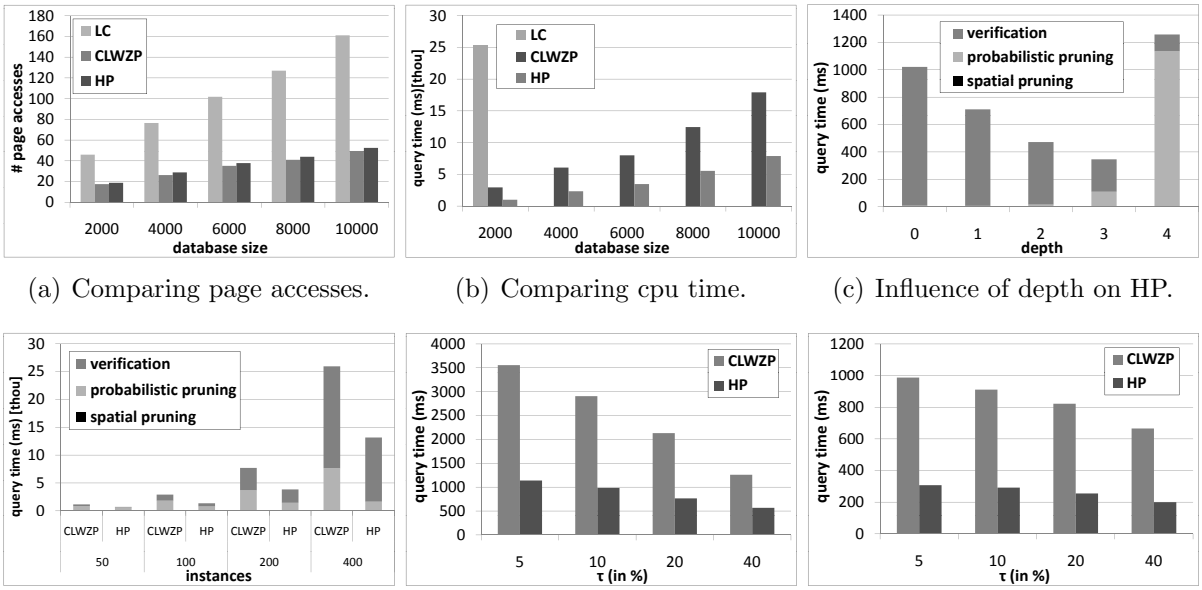
Figure 8.5: Comparison of different pruning techniques.

8.9.1 Spatial Pruning

The first set of experiments compares the spatial pruning techniques from the three competing algorithms and the MinDist/MaxDist based pruning (cf. Section 8.4) called *MinMax*. We generated 2000 uncertain 3-D objects uniformly distributed in the $[0, 1]^3$ space. Each object consists of 100 instances. 100 RNN queries were issued and we compared the number of candidates which were left after the spatial pruning step for the competing techniques. For the experiments shown in Figure 8.5(a), we tested the influence of different extensions in the dimensions of the objects. Three cases were compared: Random (random extension in each dimension, with fixed maximum), Equal (each dimension had the same extension = hypercube) and Dominant (one dimension has 5 times higher extension than the others). The results show that the spatial pruning technique from our algorithm has the highest pruning power in almost all settings. Only for objects equally extended in each dimension the LC-pruning is competitive. This is due to the reason that in this case, a sphere is a tight approximation for an uncertain object. However, in the other settings the LC-pruning yields the worst performance. We also compared the spatial pruning techniques for data with different dimensionality. Figure 8.5(b) illustrates that the advantage of HP-pruning is stable over varying dimensionality.

8.9.2 I/O-Cost

Next, we investigated the number of page accesses of the three algorithms needed on the global R*-tree (the index organizing the uncertain object approximations). The results are shown in Figure 8.6(a) for synthetic data with different database size. The LC algorithm needs by far the most page accesses which is reasonable, due to the handicaps mentioned in Section 8.4.5. The CLWZP algorithm performs even slightly better than the HP algorithm. The reason is mainly founded by the step to get the influence objects for each candidate. CLWZP here performs a search based on all instances of a candidate which produces a tighter bound than only using the approximation of the candidate (as performed in HP).



(a) Comparing page accesses. (b) Comparing cpu time. (c) Influence of depth on HP.

(d) Effect of sample size. (e) Synthetic Data. (f) Real Data.

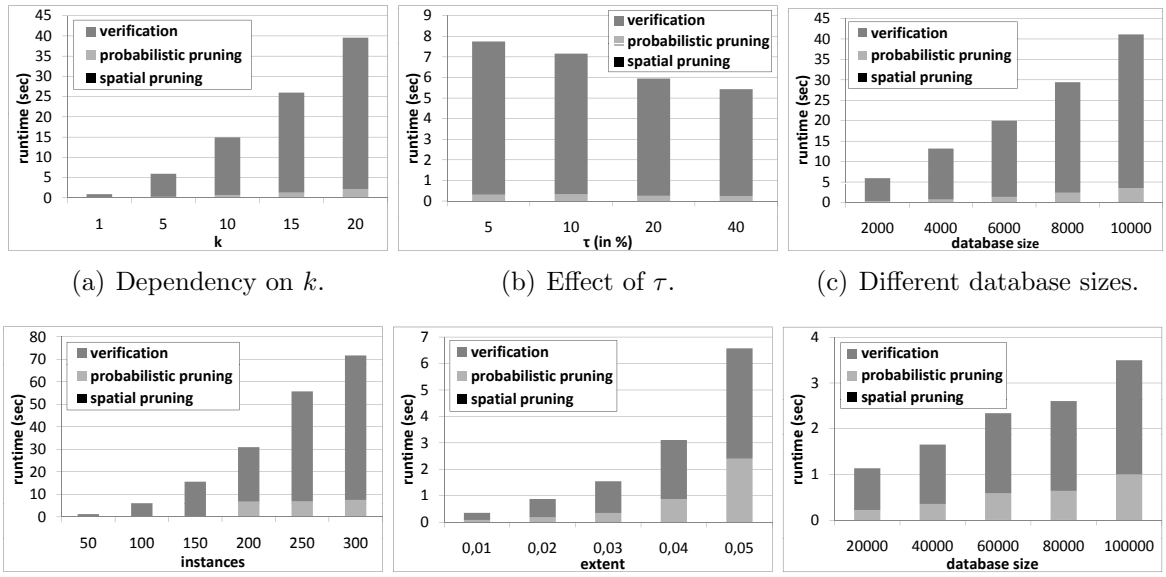
Figure 8.6: Performance of the PRNN Algorithm

The drawback of this tighter bound is a much higher computational effort, which can be seen in the experiments in the next section. In summary, even for a small page size of 1 kB, the main bottleneck of a probabilistic RNN query are the CPU-cost, not the I/O-cost.

8.9.3 CPU-Cost

The Parameter *depth***:** The main tuning parameter of the HP algorithm is *depth*. This parameter offers a tradeoff between the computational overhead for the probabilistic pruning and the verification step. This behavior can be seen in Figure 8.6(c), where by increasing *depth*, it is possible to dramatically reduce the CPU-cost in the verification step and thus the overall costs. It can be observed that for a low value of *depth*, the verification step is the main bottleneck. This is clear, since for a low value of *depth*, the set of partitions of each object X that is used for the probabilistic pruning is very small and contains large partitions. On the one hand, a small number of partitions leads to a very fast probabilistic pruning step, since only a small number of combinations of partitions has to be considered. On the other hand, the pruning power of the probabilistic pruning step is low in this case, due to the coarse approximations. The effect of the *depth* parameter in this setting is typical: there exists an optimal depth value $depth_{opt}$. Our experiments have shown that $depth_{opt}$ correlates to the height H of the R*-Tree. In all of our experiments, choosing $depth = H/2$ has shown to be a good heuristic. Determining better heuristics to find $depth_{opt}$ is part of our future work.

Scalability Experiments: Figure 8.6(b) shows the effects of the database size on the runtime of the LC, CLWZP and HP algorithms. It can be observed that the LC algorithm



(a) Dependency on k. (b) Effect of τ. (c) Different database sizes.

(d) Increasing sample size. (e) Varying extents. (f) Varying dbsize.

Figure 8.7: Behaviour of the PRkNN-Algorithm

has an extremely high CPU-cost. The reason is that the LC algorithm was proposed for continuous uncertain objects and thus requires to consider the set of all possible worlds in the verification step, which is exponential in the number of influence objects. In contrast, both CLWZP and HP scale linear in the database size. Regarding the effect of the number of instances size S, Figure 8.6(d) shows that the both algorithms CLWZP and HP scale super-linearly. The reason is that both algorithms require to sort instances of a set of influence objects in the verification step, leading to a leading to a runtime of $O(S \cdot log(S))$.

Impact of τ: Figures 8.6(e) and 8.6(f) show that the runtime of the HP algorithm decreases for an increasing value of τ for both synthetic and real datasets. The rationale is that a high value of τ reduces the minimal probability $1 - \tau$ required for an uncertain object $A \in \mathcal{DB} \setminus B$ to prune B.

PRkNN: We augmented our algorithm to answer PRkNN queries as proposed in Section 8.8 and evaluated the performance of this algorithm on the synthetic dataset using default parameters (cf. Table 8.1). In the first experiment (cf. Figure 8.7(a)), we varied the parameter k. It can be observed that the runtime scales slightly worse than linearly, which can be explained by the usage of uncertain generating functions that show a complexity of $O(k^2)$ ([22]). This is notable, since naive approaches need to consider all $\binom{N}{k}$ possible results. In the remaining experiments (Figures 8.7(b) to 8.7(d)) we evaluated the impact of the parameter τ, the database size and the number of instances per object (when setting $k = 5$). Each of these parameters scales equivalently to the $k = 1$ case. Note that in the experiments shown in Figure 8.7(d) we set $maxdepth = 3$ for more than 150 instances, matching our intuition of setting $maxdepth = H/2$ (cf. Section 8.9.3). Figure 8.7(e) and Figure 8.7(f) illustrate additional experiments on synthetic datasets focusing

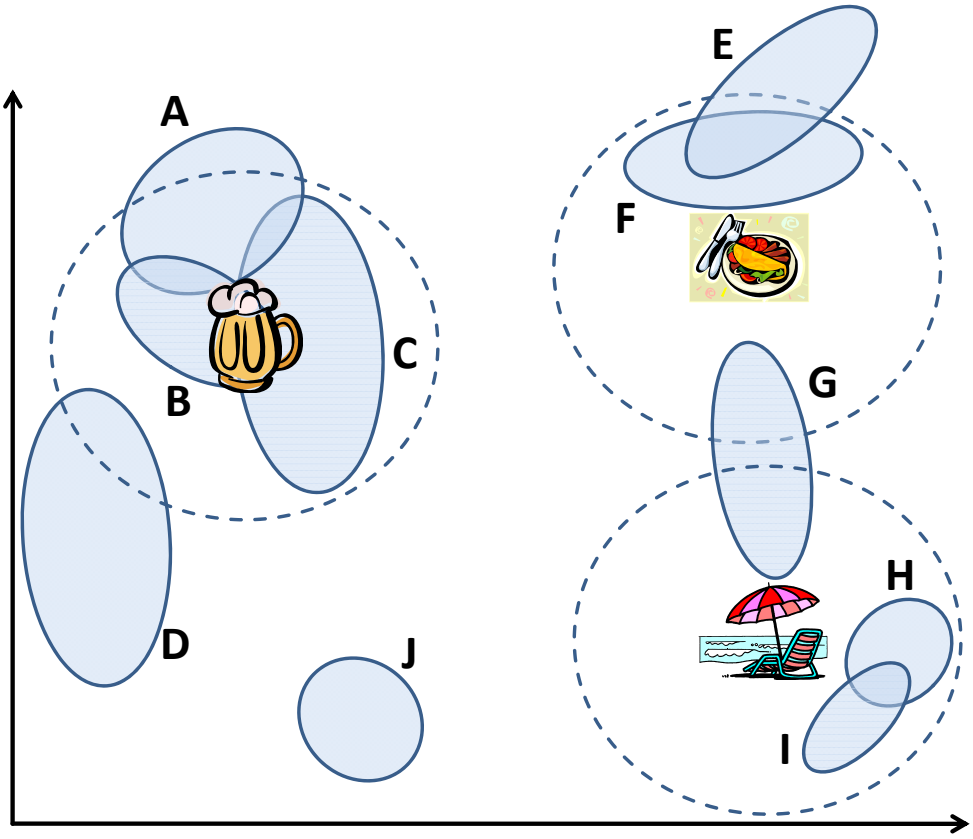
on the extent of the uncertain objects and much larger databases. The smaller the extent of the uncertain objects the more efficient queries can be performed (cf. Figure 8.7(e)). For the experiment shown in Figure 8.7(f), we lowered the average extent of the objects to 0.01. For this extent, the query times are still very good, even for very large datasets (100000 objects * 100 instances = 10 million data points).

8.10 Conclusions

In this chapter, we developed a general framework for probabilistic reverse nearest neighbor queries on uncertain data. We showed how two existing approaches and our new algorithm fit into the framework. Through new techniques to improve important parts of the framework, our algorithm is able to outperform the existing approaches under various settings. In addition, we proposed an efficient extension for probabilistic reverse k-nearest neighbor queries. For future work, we want to evaluate techniques to dynamically adapt the parameter *depth* in the probabilistic pruning step to the distribution of instances within an object.

Part IV

Mining Spatial Co-locations in Uncertain Spatial Data



This part of this thesis presents efficient solutions for the the problem of spatial co-location mining in uncertain data. A motivation of the problem, a formal definition, related work and further preliminaries are found in Chapter 9. This chapter will show that the main challenge of frequent co-location mining in uncertain data is the problem of finding the distribution of the support of a co-location. It will be shown how this problem can be mapped to the problem of probabilistic frequent itemset mining. For the problem of probabilistic frequent itemset mining, two efficient solutions are proposed:

- The first approach, presented in Chapter 10, applies the paradigm of equivalent worlds (c.f. Chapter 3) to efficiently find all co-locations in an uncertain spatial database having a non-zero probability to be frequent. Since in practice, the number of co-locations having a non-zero probability to be frequent can be very large, adaptions of this approach are presented that can efficiently return all spatial co-locations having a sufficiently high probability to exist. This approach is equivalent to applying a probabilistic threshold query predicate, as explained in Section 2.5.1. Furthermore, an approach is presented to compute and return results sorted by their probability to exist. This approach corresponds to an incremental probabilistic top-k query predicate as introduced in Section 2.5.2. Parts of this chapter have been published at the 15th ACM SIGKDD Conference On Knowledge Discovery and Data Mining (KDD) 2011 ([28]) as a full paper. While this paper presents solutions for frequent item-set mining in uncertain transaction databases, this chapter adapts these techniques to solve the problem of spatial co-location mining.

- Since a linear scan may be still be prohibitive in a very large database setting that require real-time responses, approximate approaches are presented in Chapter 11 running in constant time. Three types of approximation techniques are presented, which are based on the central limit theorem, the law of large numbers, and the law of small numbers, respectively. Theoretical and experimental evaluations show the advantages and disadvantages of each approximation approach, and research cases and applications where each approximation technique is preferable. Parts of this chapter have been published in the journal "Knowledge and Information Systems" (KAIS) 2012 ([21]) as a regular paper.

Chapter 9

Preliminaries

The task of *data mining* is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown interesting patterns such as groups of data records (cluster analysis), unusual records (outlier detection) and dependencies (association rule mining). It is the advanced analysis step of the *Knowledge Discovery in Databases* (KDD) process, which further includes preprocessing steps such as data selection, data cleaning and data transformation, as well as the evaluation of the results of the data mining step. Analogously, *spatial data mining*, is the process of discovering interesting and previously unknown, but potentially useful patterns from large spatial data sets.

This thesis will present a solution to the problem of spatial co-location mining in the context of uncertain data. To give an intuition of this problem, consider the following data mining tasks

I In an environmental monitoring application, occurrences of certain types of plants and animals have been recorded. The task is to find groups of types of plants and animals that are (not) commonly found in close vicinity. This gives information about plants and animals that live in (dis-)harmony or symbiosis, such as plants requiring disjunctive (identical) minerals from soil.

II In a geo-social network application, the task may be to automatically find groups of friends, without any apriori knowledge about their friendship graph. If a group of people is frequently found in very close vicinity of each other, on a series of Friday nights, then this group of individuals may likely be friends, or at least, share interest in the same type of Friday night activities. This information can be used to improve the geo-social network by adding missing links, to recommend similar places this group may find interesting, or to advertisement a special deal offered in a location this group may be interested in.

Both applications lead to the problem of spatial co-location mining. To motivate the problem of spatial co-location mining an example is given, followed by a formal definition of spatial co-location mining in this chapter.

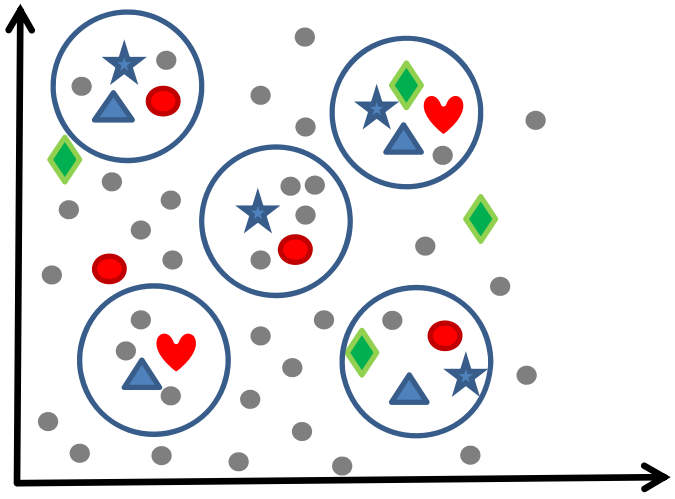


Figure 9.1: Data set for spatial co-location mining.

9.1 Spatial Co-location Mining on Certain Spatial Data

Spatial features describe the presence or absence of geographic object types at different locations. Examples of spatial features include plant species, animal species, road types, cancers, crime, and business types, or features of individuals, such as personal preferences, or simply their id. A spatial co-location pattern represents a subset of spatial features whose instances are frequently located in a spatial neighborhood. For example, "botanists may have found that there are orchids in 80% of the area where the middle-wetness green-broad-leaf forest grows" (example taken from [197]). Spatial co-location patterns may yield important insights for many applications. For example, a mobile service provider may be interested in services frequently requested by geographical neighbors, and thus gain sales promotion data. Other application domains include Earth science, public health, biology, transportation and geo-social networks.

Example 21. *Figure 9.1 shows a set of spatial objects. Each spatial object may e.g. correspond to an individual of a geo-social network. Each object may have a spatial type, called* spatial feature*, represented by the shape of each object. In general, a spatial object may have more than one spatial feature, but in this example, each uncertain object has at most one spatial feature. In this example, the spatial features are triangles, stars, diamonds, circles and hearts. Each shape depicted in Figure 9.1 corresponds to a spatial object having a spatial feature corresponding to the shape. Grey dots correspond to spatial objects having no spatial feature. In a concrete application, spatial features can, for example correspond to members of certain groups. For example, each shape may correspond to fans of the same football club, or correspond to a clique of friends. In the later example, the task may be, to find interactions between cliques of friends, i.e., to find cliques whose members often co-locate, i.e., hang out together. The circles depicted in this example represent neighborhoods, which define the set of objects which we consider to be co-located. Such a neighborhood can*

be defined, e.g., by point of interest such as bars and restaurant. For example, in the top left circle, a star, a triangle and a circle are co-located in the same neighborhood. These three objects are an instance of the set of spatial features $\{star, circle, triangle\}$*. A set of spatial features is called a co-location. The task of frequent spatial co-location mining is to find co-locations whose instances are co-located frequently. In this example, the co-location* $\{star, circle\}$ *and the co-location* $\{star, triangle\}$ *can be considered frequent, as there is three instances where these co-locations appear together. This number of occurrences of a co-location* S *is called support* $supp(S)$ *of* S*. The minimal support that a set requires in order to be considered frequent is a user specified parameter called minSup. In this example, for* $munSup = 3$*, the co-locations* $\emptyset$*,* $\{star\}$*,* $\{circle\}$*,* $\{triangle\}$*,* $\{star, circle\}$ *and* $\{star, triangle\}$ *are the only frequent co-locations, returned by a frequent co-location mining algorithm. Trivial co-locations having less than two features are usually omitted.*

To formally define the problem of spatial co-location mining, we first have to define the concept of a spatial co-location.

Definition 34 (Spatial Co-location Instance)**.** *Given a reflexive and symmetric neighbor relation* $\mathcal{R}$ *over a spatial database* $\mathcal{DB}$*, a* spatial co-location instance *is a set* $I \subseteq \mathcal{DB}$ *of spatial objects that form a clique [20] under the relation* $\mathcal{R}$*, i.e.,* $\forall o_1, o_2 \in I : (o_1, o_2) \in R$*.*

The definition of neighbor relation $\mathcal{R}$ is an input and should be based on the semantics of the application domains. The neighbor relation $\mathcal{R}$ may be defined using metric relationships such as Euclidean distance or shortest-path distance in a graph such as a road network.

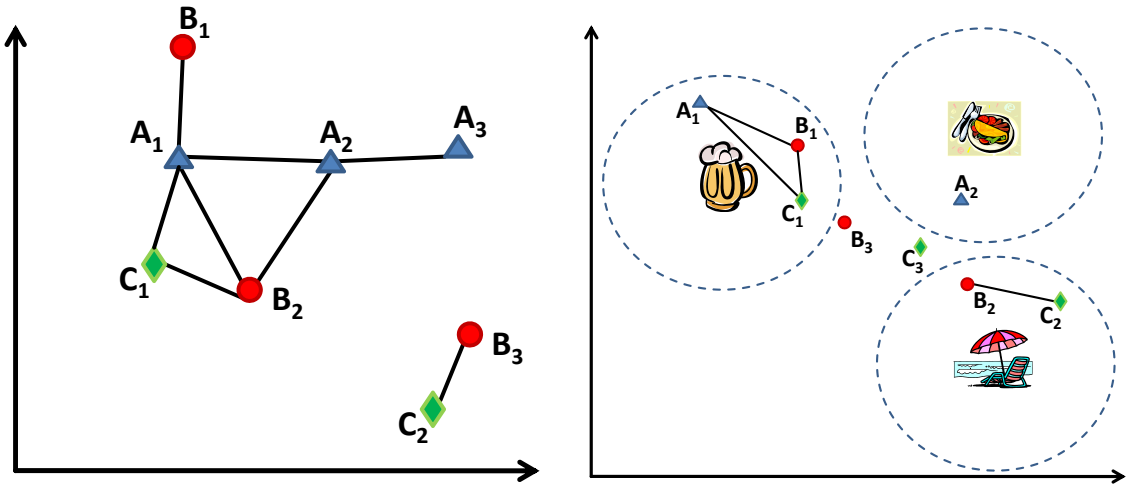
Within this thesis, a neighbor relation will be used that is particularly important in social network applications. This relation uses a set of interesting spatial locations, such as bars, restaurants and football stadiums. Two individuals are co-located if they are sufficiently close to the same location. Formally,

Definition 35. *Let* $\mathcal{L}$ *be a set of spatial locations, and let* $\mathcal{DB}$ *be a database of spatial objects. The neighbor relation* $\mathcal{R}$ *is defined as follows*

$$(o_1, o_2) \in \mathcal{R} \Leftrightarrow \exists l \in \mathcal{L} : dist(o_1, l) \leq \epsilon \wedge dist(o_2, l) \leq \epsilon$$

Definition 36 (Frequent Spatial Co-location Mining)**.** *Given a set of spatial features* $\mathcal{F} = \{f_1, ..., f_k\}$ *of* k *spatial features, given a database* $\mathcal{DB} = \{o_1, ..., o_N\}$ *of* N *spatial objects each having a set* $f(o_i \in \mathcal{DB}) \subseteq F$ *of spatial features, and given a positive integer minSup, a frequent spatial co-location mining algorithm returns all sets* $S \subseteq F$ *of features such that there exists at least minSup spatial co-locations instances* I *such that* $S \subseteq \bigcup_{o \in I} f(o)$*.*

Since a single database object may have more than a single spatial feature, a single object o may cause a co-location of features $f(o)$, or a set of two objects may define a co-location instance containing more than two spatial features. This general definition is meaningful in application such as Application I, where spatial features may correspond to diseases carried by animals in plants. A single poor animal or plant may clearly be affect



(a) Distance Based Neighborhood Semantics (b) Location Based Neighborhood Semantics

Figure 9.2: Spatial co-location mining in certain spatial data.

by more than one disease at a time, and a resulting co-location of diseases that may be relevant, independent of whether this set of diseases is carried by a single specimen or by a group. In contrast, in an application such as Application II, where spatial features may correspond to the affiliation of scientists, a single individual may have multiple affiliations, for example due to this researcher currently being on sabbatical at a different university. In the later case, an addition constraint may be imposed on Definition 36 to ensure that co-location instances correspond to distinct object sets.

The neighborhood relation used in Example 21 is chosen very arbitrarily, and for illustration purpose only. Clearly, the choice of neighbor relation, i.e., the definition of the predicate that a spatial object is required to satisfy in order to belong to an instance of a co-location, is very important but highly application specific. In an application such as described above in Application I, two objects are usually considered neighbors if the distance between these object does not exceed some predefined threshold. A set of objects is co-located, if all pairs of these objects are neighbors, i.e., if they form a clique in the corresponding neighbor relation. These semantics of co-location are used in most related work (e.g. [214, 93, 200, 167, 92, 197]). An example of such state-of-the-art semantics is depicted in Figure 9.2(a). Here, the spatial neighborhood relation is illustrated by edges connecting spatial objects that are sufficiently close to each other. In this example, there is a number of instances of co-locations: For example, one instance of a co-location contains objects $\{A_1, B_2, C_1\}$, since all these objects are pairwise connected. This co-location contains the spatial features $\{triangle, diamond, circles\}$.

Such a definition of spatial neighborhood may be inappropriate for other applications such as Application II given above. In this application, the neighborhood relation is defined by objects in proximity (i.e., sufficiently close to) an interesting location, such as a bar, a restaurant or the beach. An example is given in Figure 9.2(b) where objects represent

individuals, of a geo-social network. We can see that individuals A_1, A_2 and A_3 are having a good time together in the bar; B_2 and C_2 are enjoying the beach; while A_2 is having dinner in solitude. Unlike in an application such as Application I, spatial neighborhood between two objects is not defined by the distance between the objects, but rather by the existence of an interesting location in proximity of both objects.

The later semantics of spatial neighborhood will be used in this part of this thesis, due to its applicability to geo-social network data. For example, the spatial features (triangles, diamonds and circles) depicted in Figure 9.2(b) may represent the individual's affiliations. Frequent co-locating mining on such data may yield interesting patterns, such as "Members of LMU and HKU are frequently to be found at the same location, while members of TUM are often found in solitude or among themselves". While the automatic extraction of such patterns from large sets of data may be interesting in some applications, the true power of these semantics shows if we consider the temporal component of data. If we consider time, each spatial feature may correspond to an individual. Thus, spatial objects having the same spatial features correspond to the same individual, but at different points of time. In such an application, Figure 9.2(b) corresponds to the events that all three individuals corresponding to triangles, diamonds and circles, are found together in the bar at one time, while at a later time they split up between the beach and the restaurant. In such an application, each instance of a co-location corresponds to a (location l , time t) pair, i.e., each instance of a co-location corresponds to the set of individuals that have been at the same location l at the same time t. The big challenge here is that the temporal dimension leads to very large sets of co-location data, since every location and time pair leads to a possibly non-empty co-location instance. This type of data is already being collected in vast amounts every day in geo-social networks, where users can user the check-in function to submit their current location to the geo-social network. Analyzing this data will allow to find groups of people that significantly often share the same location. Significance tests are required to exclude incidental co-locations (i.e., the case where individuals happen to share a location by chance, without actually meeting) with a high probability. The resulting groups can be used to improve the underlying social network, e.g., by adding missing links, or by allowing the social network to recommend locations and events, tailored to the preferences of the group.

In traditional co-location mining, the location of an object is known for certain. Under this assumption, a lot of work has been published in the last decade [214, 93, 200, 167, 92, 214]. A survey on the field of co-location mining on certain spatial data be found in [134, 135]. However, in many real applications such as plant disease diagnosis, environmental surveillance and geo-social networks, the location of objects is uncertain. In the following, the problem of probabilistic spatial collocation mining on uncertain spatial data is defined.

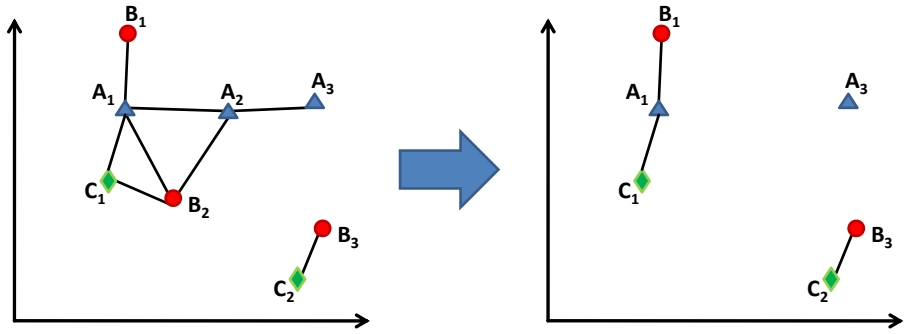


Figure 9.3: A possible world using the neighborhood relation of [197] .

9.2 Spatial Co-location Mining on Uncertain Spatial Data

For example, an epiphytology[1] expert may highly suspect (but cannot guarantee) that a plant suffers from mildew (a plant disease). The uncertainty of such a suspicion can be expressed by existential probability. In this spatially uncertain data, each disease is a feature and a potential plant with the disease is an instance of the feature and is associated with an existential probability expressing the likelihood of this plant having the disease. For instance, at longitude 97.7 E and latitude 26.5 N in "Three Parallel Rivers of Yunnan protected Areas" area, rhododendra have a 70likelihood of having blight (example taken from [197]). Using precise spatial data would mean that these plants would be considered to have 100% likelihood of having the disease. Also, in geo-social networks, the check in position of a user is inherently uncertain, for many reasons described in the introduction (Part I) of this thesis.

Only recently, the research community has tackled the challenge of spatial co-location mining in uncertain data. Recent work ([197]) considers existential uncertainty in spatial data. In this model, each object has a probability to be present in the database. Thus, in each possible world, the set of nodes of the corresponding neighbor graph is different. An example possible world for the neighbor graph of Figure 9.2(a), is shown in Figure 9.3. Here, the non-existent nodes corresponding to objects A_2 and B_2 are removed. It can be seen that the graph structure has changed significantly, and it seems intuitive, that in practice, possible worlds may be highly heterogeneous. For this reason, to the best of our knowledge, no efficient solution for the problem of frequent co-location mining in uncertain data has been present. In particular, the solution of [197] has a run-time polynomial in the number of possible worlds, thus exponential in the number of uncertain objects. The reason for this high complexity is the neighborhood relation used in [197], which is chosen arbitrary, i.e., this approach can be applied to any neighborhood relation. This fact makes efficient co-location mining hard: For example, assume the neighborhood relation R between two objects is defined by the predicate of having a distance less than some

[1]The study of plant diseases.

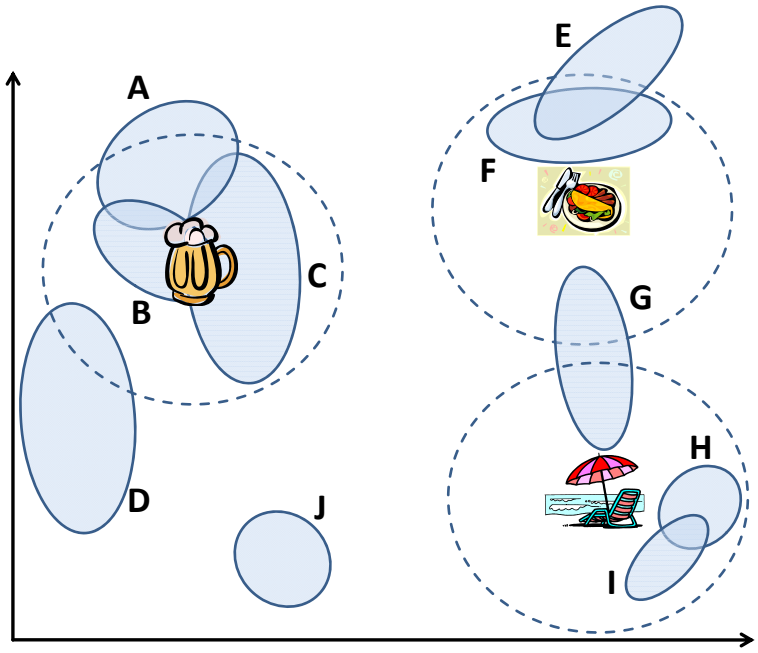


Figure 9.4: Example Uncertain Spatial Databases with points of interest.

threshold, corresponding to Application I described above. For three uncertain objects A, B and C, the predicates $R(A, B)$ and $R(B, C)$ are stochastically dependent, despite the assumption of independence between objects. The reason is that both predicates depend on the same random variable corresponding to the location (and existence) of B. This *distance dependency* is described in more detail in Example 16 in Chapter 4.

This is the reason why in this part of this thesis, an explicit neighborhood function is given in Definition 35, which is necessary to allow efficient frequent co-location mining. Unlike the work of [197], both existential and attribute uncertainty are allowed in the remainder of this chapter.

Table 9.1: Co-locations corresponding to Example 22 and Figure 9.4.

Location	**Time**	**(User, Probability)***
Bar	8:00 p.m.	$(A, 0.8); (B, 1); (C, 1); (D, 0.1))$
Restaurant	8:00 p.m.	$(E, 0.4); (F, 1); (G, 0.4)$
Cinema	8:00 p.m.	$(G, 0.4); (H, 1); (I, 1);$
Bar	9:00 p.m.	$(D, 0.6); (E, 1); (F, 0.3)$
Restaurant	9:00 p.m.	$(B, 0.4); (C, 1); (D, 0.4)$
Cinema	9:00 p.m.	$(I, 1)$
...	....	...

An example of the problem of frequent co-location mining in uncertain spatial data using the neighborhood relation of Definition 35 is given in the following.

Example 22. *Consider uncertain positions of individuals in a geo-social network application. The task is to find groups of people that commonly spend time at the same locations, in order to predict missing social links in the underlying social network, or in order to direct special deals to such groups. Figure 9.4 exemplarily shows the position of a individuals $A, ..., J$, together with three locations: a bar, a restaurant and a cinema. For simplicity, each of these locations is represented by a circular region, but other representations such as polygons can be used. Due to the uncertainty of the location of individual users. For example, it is not possible to tell for certain, whether user A is located inside the bar, or just barely outside of it. In contrast, users B and C are certainly inside the bar, while user J is certainly not in the bar. The probability $P(U \text{ in } l)$ that a user U is located inside a location l can be computed using techniques presented in Chapter 4.*[2] *Exemplary probabilities $P(U \text{ in } l)$ for all users U and all locations l are shown in Table 9.1. In this table, the time stamp 8:00 p.m. corresponds to the setting of Figure 9.4. At this time, the users, A, B and C are co-located at the bar with a high probability. However, at time 9:00 p.m., user A is no longer located with users B and C, who have moved to the restaurant. In contrast, it is likely that the group consisting of users E, F and G remained together, hitting the bar after the restaurant.*

Clearly, the number of co-locations may be extremely large, since in an application like this, there may be one non-empty co-location for each combination of time stamp and location. The task of probabilistic co-location mining is to find groups of users (objects), having a significantly high probability of having spent time at the same location for a sufficiently large number of times.

[2]For the case proximity to a location is not modelled by a circle, an adaption of the techniques in Chapter 4 can be made easily, by replacing distance calculation by intersection tests between points and polygons.

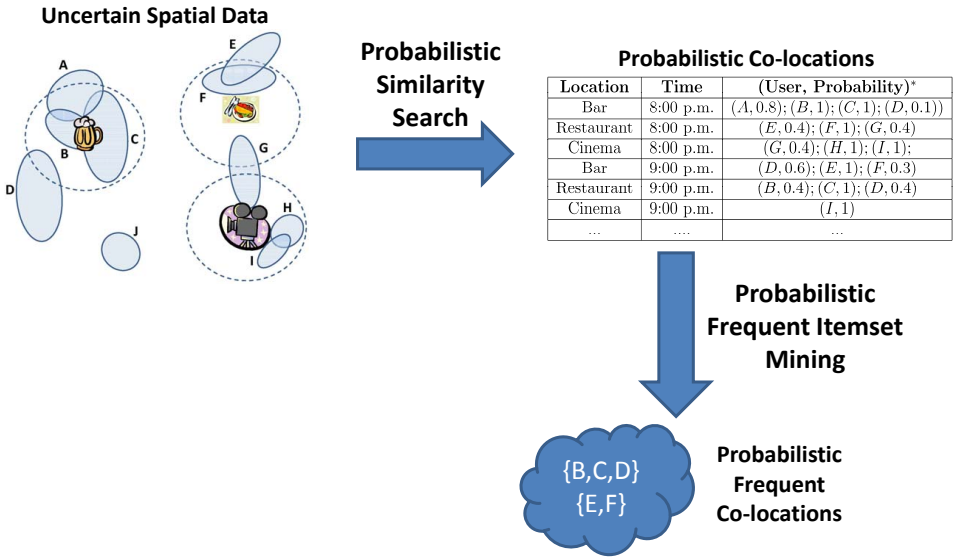


Figure 9.5: Workflow of probabilistic spatial co-location mining.

9.2.1 Problem Definition

In a nutshell, the problem of probabilistic co-location mining requires two subtasks to be solved, as illustrated in Figure 9.5:

- First, for each location l and each time interval t, probabilistic instances have to be computed and derived. This requires to compute the probabilities of all objects, to be close to location l at time t. This task requires to utilize probabilistic similarity search methods on uncertain spatial data to derive the probability that a given object is a member of a co-location instance. For the neighbor relation given in Definition 35, this step requires to perform probabilistic range queries, using the locations $\mathcal{L}$ as query points. The problem of probabilistic similarity search on uncertain spatial data has been covered thoroughly in Part III of this thesis. Therefore, the focus of this part will directed to the second subtask. As a result of the first step, an uncertain spatial database is transformed into a probabilistic co-location database such as the one depicted in Table 9.1.

- Second, all probabilistic co-location instances need to be mined in order to detect subsets of spatial features having a statistically significantly high probability to be co-located frequently in the database. For this subtask, we can assume that a database $\mathcal{DB}$ of probabilistic co-locations such as featured in Table 9.1 is given as a result of solving the first subtask. Given such a database, the task of finding probabilistic frequent co-locations in such a database is equivalent to the problem of probabilistic frequent itemset mining ([28]) in uncertain transaction data. Both problems, of probabilistic mining of spatial co-locations in uncertain spatial data, as well the the problem of probabilistic frequent itemset mining in uncertain transaction data, are formally defined in the following.

Definition 37 (Uncertain Co-Location Database)**.** *Let F be a set of spatial features. An* uncertain co-location *database $\mathcal{DB}$ is a set of probabilistic co-location instances. Each probabilistic co-location instance $T = (f \in F, P(f)) \in \mathcal{DB}$ contains a set of spatial features, each associated with a probability. For each pair $(f \in F, P(f))$, the probability $P(f)$ describes the probability that spatial feature f is present in the probabilistic co-location instance T.*

A co-location is a *frequent co-location* if it occurs in at least *minSup* co-location instances, where *minSup* is a user specified parameter. The number of instances of a co-location is denoted as the support $supp(S)$ of S. In uncertain co-location databases however, the support of a co-location is uncertain; it is defined by a discrete probability distribution function (PDF).

Definition 38 (Probabilistic Support)**.** *Let $\mathcal{DB}$ be an uncertain co-location database and let $X \subseteq \mathcal{F}$ be a set of spatial features. The support of X is a probability density function*

$$supp(X) : IN_0 \to [0, 1]$$
$$n \mapsto P(supp(X) = n).$$

that maps each non-negative integer n to the probability that the support of features X equals n.

Therefore, each set of spatial features has a *frequentness probability* [3] – the probability that it is frequent.

Definition 39 (Frequentness Probability)**.** *Let $\mathcal{DB}$ be an uncertain co-location database, let $X \subseteq \mathcal{F}$ be a set of spatial features and let minSup be a non-negative integer. The probability*

$$P(supp(X) \geq minSup) = \sum_{i=minSup}^{\infty} P(supp(X) = i) = 1 - \sum_{i=0}^{minSup-1} P(supp(X) = i).$$

is called frequentness probability *of X.*

Given the frequentness probability of a set of spatial features X, we can apply a probabilistic threshold query predicate in order to decide whether the frequentness probability of X is significantly high.

Definition 40. *A* Probabilistic Frequent Set of Spatial Features *is a set of spatial features with a* frequentness probability *of at least τ.*

We are now able to specify the *Probabilistic Frequent spatial co-location mining problem* as follows; Given an uncertain spatial database $\mathcal{DB}$, a minimum support scalar *minSup* and a frequentness probability threshold τ, find all probabilistic frequent sets of spatial features. The parameter τ is the user specified minimum confidence in the frequentness of a set of spatial features. If τ is set close to 1 then the user is interested only in very confident results, while a small value for τ (close to 0) would lead to results that also include itemsets which are not frequent in most possible worlds.

[3] Frequentness is the rarely used word describing the property of being frequent.

9.2.2 Probabilistic Frequent Itemset Mining

The definition of a uncertain co-location database is equivalent to the definition of an uncertain transaction database defined in [28].

Definition 41 (Uncertain Transaction Database)**.** *Let I be a set of items. An* uncertain transaction *database $\mathcal{T}$ is a set of probabilistic transactions. Each transaction $T = (i \in I, P(i)) \in \mathcal{T}$ contains a set of items, each associated with a probability. For each pair $(i \in I, P(i))$, the probability $P(i)$ describes the likelihood that i is present in the probabilistic transaction T.*

This equivalence between Definition 37 and Definition 41 allows to interpret the problem of probabilistic frequent co-location mining in uncertain spatial data, as the problem of probabilistic frequent itemset mining in uncertain transaction data. This can be done be the following interpretation:

- a spatial feature is interpreted as an item
- a probabilistic co-location instance is treated as a transaction.

Using this interpretation, a probabilistic item and an uncertain transaction are defined as follows.

Definition 42 (Probabilistic Item)**.** *Let T be a transaction database. A* probabilistic item *is an item $x \in I$ whose presence in a transaction $t \in T$ is defined by an* existential probability *$P(x \in t) \in (0, 1)$. A* certain item *is an item where $P(x \in t) \in \{0, 1\}$. I is the set of all possible items.*

Definition 43. *An* uncertain transaction *t is a transaction that contains uncertain items. A transaction database T containing uncertain transactions is called an* uncertain transaction database*.*

Definition 44. *A* Probabilistic Frequent Itemset *(PFI) is an itemset with a* frequentness probability *of at least τ.*

This interpretation allows to directly apply solution for the problem of probabilistic frequent itemset mining on uncertain data, which are presented in the following two chapters. The sets of frequent items returned by such algorithms can then be interpreted as sets of spatial features. These sets of spatial features correspond to spatial co-locations by Definition 34.

Chapter 10

Probabilistic Frequent Itemset Mining

Location	Time	Person	Prob.
Bar	t_1	Andy	1.0
Bar	t_1	Brian	0.2
Restaurant	t_2	Chris	0.4
Restaurant	t_2	Brian	0.7

ID	Transaction
t_A	(Andy, 1.0) ; (Brian, 0.2)
t_B	(Chris, 0.4) ; (Brian, 0.7)

World	TransactionDB	Prob.
1	{Andy} ; {}	0.144
2	{Andy, Brian} ; {}	0.036
3	{Andy} ; {Chris}	0.096
4	{Andy, Brian} ; {Chris}	0.024
5	{Andy} ; {Brian}	0.336
6	{Andy, Brian} ; {Brian}	0.084
7	{Andy} ; {Chris, Brian}	0.224
8	{Andy, Brian}; {Chris, Brian}	0.056

Figure 10.1: Possible Worlds of an Uncertain Transaction Database.

Association rule analysis is one of the most important fields in data mining. It is commonly applied to market-basket databases for analysis of consumer purchasing behavior. Such databases consist of a set of transactions, each containing the items a customer purchased. The database is analyzed to discover associations among different items. The most important and computationally intensive step in the mining process is the extraction of *frequent itemsets* – sets of items that occur in at least *minSup* transactions. Besides market-basket analysis, frequent itemset mining is also a core component in applications such as other applications, such as association-rule mining [8] and sequential-pattern mining [9].

As an example of an uncertain transaction database, consider the following example.

Example 23. *The top left table of Figure 10.1 shows the probabilistic result of probabilistic range queries for two locations* bar *and* restaurant. *This result implies that Andy was*

located in the bar for certain at time t_1, while Brian has only a 0.2 probability of having visited the bar at the same time. At time t_2, Brian and Chris are located in the restaurant with a probability of 0.7 and 0.4 respectively. Probabilistic range queries returning such results have been presented in Chapter 4. This result is stored in an uncertain transaction database, by simply performing a GROUP BY, using attributes Location and Time. Each resulting group corresponds to a transaction and is given a unique transaction id. The resulting uncertain transaction database is given by the table in the bottom left of Figure 10.1. Since in this database, there is three uncertain items, there is a total of $2^3 = 8$ possible worlds, which are depicted in the right of Figure 10.1.

Clearly, applying a traditional frequent itemset mining algorithms to every single possible world is infeasible, as the number of possible transaction databases grows way too large for any non-trivial number of probabilistic items. In the following, solutions for the problem of probabilistic frequent itemset mining will be given. To apply these solutions to the problem of probabilistic frequent co-location mining, we can simply rewrite the problem as a probabilistic frequent itemset mining problem.

10.1 Related Work

The problem of probabilistic co-location mining in uncertain spatial data is related to the problem of frequent itemset mining in uncertain transaction databases. Existing solutions for this problem transform uncertain items into certain ones by thresholding the probabilities. For example, by treating all uncertain items with a probability value higher than 0.5 as being present, and all others as being absent in a transaction. Such an approach loses useful information and leads to inaccuracies. Existing approaches in the literature are based on expected support ([53, 54, 6]). Itemsets are considered frequent if the expected support exceeds *minSup*. Effectively, this approach returns an estimate of whether an object is frequent or not with no indication of how good this estimate is. Since uncertain transaction databases yield uncertainty w.r.t. the support of an itemset, the probability distribution of the support and, thus, information about the confidence of the support of an itemset is very important. This information, while present in the database, is lost using the expected support approach.

There is a large body of research on Frequent Itemset Mining (FIM) but very little work addresses FIM in uncertain databases [53, 54, 121]. The approach proposed by Chui et. al [54] computes the expected support of itemsets by summing all itemset probabilities in their U-Apriori algorithm. Later, in [53], they additionally proposed a probabilistic filter in order to prune candidates early. In [121], the UF-growth algorithm is proposed. Like U-Apriori, UF-growth computes frequent itemsets by means of the expected support, but it uses the FP-tree [85] approach in order to avoid expensive candidate generation. In contrast to our probabilistic approach, itemsets are considered frequent if the expected support exceeds *minSup*. The main drawback of this estimator is that information about the uncertainty of the expected support is lost; [53, 54, 121] ignore the number of possible worlds in which an itemsets is frequent. [212] proposes exact and sampling-based algorithms to find likely

frequent items in streaming probabilistic data. However, they do not consider itemsets with more than one item. To the best of our knowledge, our approach in [28] was the first that is able to find frequent itemsets in an uncertain transaction database in a probabilistic way.

However, this publication has stimulated research on the field of probabilistic mining of frequent itemsets in uncertain transaction data, creating a large number of follow up publications. A detailed survey can be found in [187]. In [195, 196], an approach is presented to approximate the support PDF of an itemset using a Poisson distribution. Approach yields a very small error if the database is sufficiently large. This approximation furthermore allows to compute the support PDF of an item much faster than the exact approach presented in [28] and in this chapter. The idea of [195, 196] is used to study a variety of approximation techniques in Chapter 11, including Expected support, Normal approximation and Poisson approximation. An approach to accelerate the computation of our approach in [28] was presented by [106], using massive parallelization exploiting GPGPU (General-Purpose computation on GPU). Furthermore, the related problem of mining frequent subgraphs over uncertain graphs [127, 219, 218] has gained alot of research interest in the last years. Finally, an approach for probabilistic frequent itemset mining on uncertain data avoiding multiple database scans incurred by the candidate generation step of [28] has been proposed by us in [29].

10.2 Probabilistic Frequent Itemsets

One simple approach is to transform an uncertain database into a non-uncertain database by setting the item probabilities to 0 or 1 and then applying a traditional frequent itemset detection method. For example, probabilities less then 0.5 could be mapped to 0 and probabilities above 0.5 could be mapped to 1. However, such a transformation obviously involves loss of information and accuracy. Furthermore, we would have no idea how confident we could be in the results. In particular, itemsets that are often associated with probabilities close to 0.5 yield a very large error in the result. Another approach is to use the probabilities associated with the itemsets in order to compute the expected support of an itemset.

Recall that previous work was based on the expected support [53, 54, 121].

Definition 45. *Given an uncertain transaction database T, the* expected support $E(X)$ *of an itemset X is defined as* $E(X) = \sum_{t \in T} P(X \subseteq t)$.

The expected support of an itemset X can be efficiently computed by a single scan over the uncertain transaction database while building the sum of all probabilities associated with the itemset X. Considering an itemset frequent if its expected support is above *minSup* has a major drawback. Uncertain transaction databases naturally involve uncertainty concerning the support of an itemset. Considering this is important when evaluating whether an itemset is frequent or not. However, this information is forfeited when using the expected support approach.

ID	Co-location
t_1	(A, 0.8) ; (B, 0.2) ; (D, 0.5) ; (F, 1.0)
t_2	(B, 0.1) ; (C, 0.7) ; (D, 1.0) ; (E, 1.0) ; (G, 0.1)
t_3	(A, 0.5) ; (D, 0.2) ; (F, 0.5) ; (G, 1.0)
t_4	(D, 0.8) ; (E, 0.2) ; (G, 0.9)
t_5	(C, 1.0) ; (D, 0.5) ; (F, 0.8) ; (G, 1.0)
t_6	(A, 1.0) ; (B, 0.2) ; (C, 0.1)

Figure 10.2: Example of an uncertain co-location database.

Example 24. *As an example, consider the database depicted in Figure 10.2, containing a set of uncertain co-location instances. Treating each co-location instance as a transaction, the expected support of the itemset* $\{D\}$ *is* $E(\{D\}) = 3.0$*. The fact that* $\{D\}$ *occurs for certain in one transaction, namely in* t_2*, and that there is at least one possible world where* D *occurs in five transactions are totally ignored when using the expected support in order to evaluate the frequency of an itemset. Indeed, suppose* $minSup = 3$*; do we call* $\{D\}$ *frequent? And if so, how certain can we even be that* $\{D\}$ *is frequent? By comparison, consider itemset* $\{G\}$*. This also has an expected support of* 3*, but its presence or absence in transactions is more certain. It turns out that the probability that* $\{D\}$ *is frequent is* 0.7 *and the probability that* G *is frequent is* 0.91*. While both have the same expected support, we can be quite confident that* $\{G\}$ *is frequent, in contrast to* $\{D\}$*. An expected support based technique does not differentiate between the two.*

The confidence with which an itemset is frequent is very important for interpreting uncertain itemsets. We therefore require concepts that allow us to evaluate the uncertain data in a probabilistic way. In this section, we formally introduce the concept of probabilistic frequent itemsets.

Notation	Description
W, w	Set of all possible worlds, Possible world instance $w \in W$
T, t	Uncertain transaction database, transaction $t \in T$
I	Set of all items
X, x	Itemset $X \subseteq I$, item $x \in I$
$S(X, w)$	Support of X in world w
$P_i(X)$	Probability that the support of X is i
$P_{\geq i}(X)$	Probability that the support of X is *at least* i
$P_{i,j}(X)$	Probability that i of the first j transactions contain X
$P_{\geq i,j}(X)$	Probability that *at least* i of the first j transactions contain X

Table 10.1: Summary of Notations of this Chapter

10.2.1 Probabilistic Support

In uncertain transaction databases, the support of an item or itemset cannot be represented by a unique value, but rather, must be represented by a discrete probability distribution. Before we give a formal definition of the *support probability distribution*, we need to define the *support probability.* This is the probability that an itemset has a certain support.

Definition 46. *Given an uncertain (transaction) database T and the set W of possible worlds (instantiations) of T, the* support probability $P_i(X)$ *of an itemset X is the probability that X has the support i. Formally,*

$$P_i(X) = \sum_{w_j \in W, (S(X,w_j)=i)} P(w_j)$$

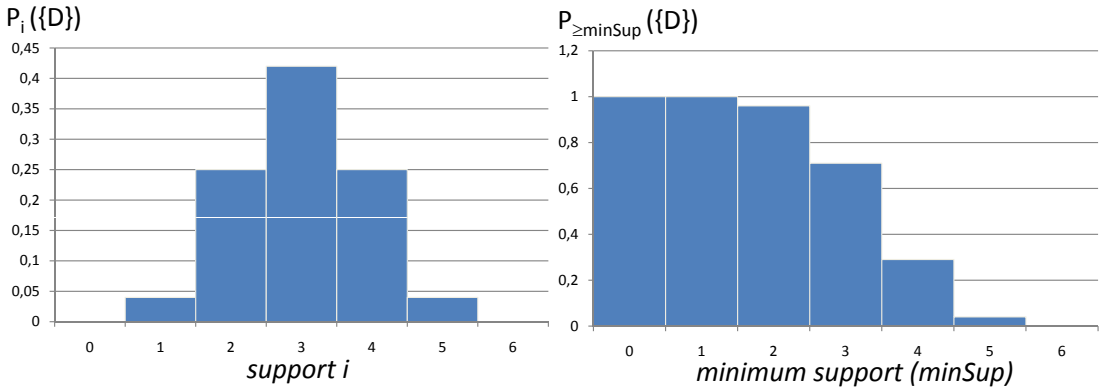
where $S(X, w_j)$ is the support of X in world w_j.

Intuitively, $P_i(X)$ denotes the probability that the support of X is exactly i . The support probabilities associated with an itemset X for different support values form the *support probability distribution* of the support of X.

Definition 47. *The probabilistic* support *of an itemset X in an uncertain transaction database T is defined by the support probabilities of X ($P_i(X)$) for all possible support values $i \in \{0, ..., |T|\}$. This probability distribution is called* support probability distribution. *The following statement holds: $\sum_{0 \leq i \leq |T|} P_i(X) = 1.0$.*

Returning to our example of Figure 10.2, Figure 10.3(a) shows the support probability distribution of itemset $\{D\}$.

The number of possible worlds $|W|$ that need to be considered for the computation of $P_i(X)$ is extremely large. In fact, we have $O(2^{|T| \cdot |I|})$ possible worlds, where $|I|$ denotes the total number of items. In the following, we show how to compute $P_i(X)$ without materializing all possible worlds.



(a) Support probability distribution of $\{D\}$ (b) Frequentness probabilities of $\{D\}$

Figure 10.3: Probabilistic support of itemset $\{D\}$ in the uncertain database of Figure 10.2.

Lemma 29. For an uncertain transaction database T with mutually independent transactions and any $0 \leq i \leq |T|$, the support probability $P_i(X)$ can be computed as follows:

$$P_i(X) = \sum_{S \subseteq T, |S|=i} \left(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t)) \right) \tag{10.1}$$

Note that the transaction subset $S \subseteq T$ contains exactly i transactions.

Proof. The transaction subset $S \subseteq T$ contains i transactions. The probability of a world w_j where all transactions in S contain X and the remaining $|T - S|$ transactions do not contain X is $P(w_j) = \prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t))$. The sum of the probabilities according to all possible worlds fulfilling the above conditions corresponds to the equation given in Definition 46. □

10.2.2 Frequentness Probability

Recall that we are interested in the *probability* that an itemset is frequent, i.e. the probability that an itemset occurs in at least *minSup* transactions.

Definition 48. *Let T be an uncertain transaction database and X be an itemset. $P_{\geq i}(X)$ denotes the probability that the support of X is* at least *i, i.e. $P_{\geq i}(X) = \sum_{k=i}^{|T|} P_k(X)$. For a given minimal support* minSup $\in \{0, \ldots, |T|\}$*, the probability $P_{\geq \text{minSup}}(X)$, which we call the* frequentness probability *of X, denotes the probability that the support of X is at least* minSup*.*

Figure 10.3(b) shows the frequentness probabilities of $\{D\}$ for all possible *minSup* values in the database of Figure 10.2. For example, the probability that $\{D\}$ is frequent when $minSup = 3$ is approximately 0.7, while its frequentness probability when $minSup = 4$ is approximately 0.3.

The intuition behind $P_{\geq minSup}(X)$ is to show how confident we are that an itemset is frequent. If the probability that the support is above *minSup* is high, we can be fairly confident that the itemset is indeed frequent. With this policy, the frequentness of an itemset becomes subjective and the decision about which candidates should be reported to the user depends on the application. Hence, we use the minimum frequentness probability τ as a user defined parameter. Some applications may need a low τ, while in other applications only highly confident results should be reported (high τ).

In the possible worlds model we know that $P_{\geq i}(X) = \sum_{w_j \in W:(S(X,w_j)\geq i)} P(w_j)$. This can be computed according to Equation 10.1 by

$$P_{\geq i}(X) = \sum_{S \subseteq T, |S| \geq i} (\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t))). \tag{10.2}$$

Hence, the frequentness probability can be calculated by enumerating all possible worlds satisfying the *minSup* condition through the direct application of Equation 10.2. This naive approach is very inefficient however. We can speed this up significantly. First, note that typically $minSup << |T|$ and the number of worlds with support i is at most $\binom{|T|}{i}$. Hence, enumeration of all worlds w in which the support of X is greater than *minSup* is much more expensive than enumerating those where the support is less than *minSup*. Using the following easily verified Corollary, we can compute the frequentness probability exponentially in $minSup << |T|$.

Corollary 9. $P_{\geq i}(X) = 1 - \sum_{S \subseteq T:|S|<i} (\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T-S} (1 - P(X \subseteq t)))$.

Despite this improvement, the complexity of the above approach, called **Basic** in our experiments, is still exponential w.r.t. the number of transactions. In Section 10.3 we describe how we can reduce this to linear time.

10.3 Efficient Computation of Probabilistic Frequent Itemsets

This section presents the Poison-binomial recurrence based approach, which avoids the enumeration of possible worlds in calculating the frequentness probability and the support distribution. We also present probabilistic filter and pruning strategies which further improve the run time of this method.

10.3.1 Efficient Computation of Probabilistic Support

The key to our approach is to consider it in terms of sub-problems. First, we need appropriate definitions;

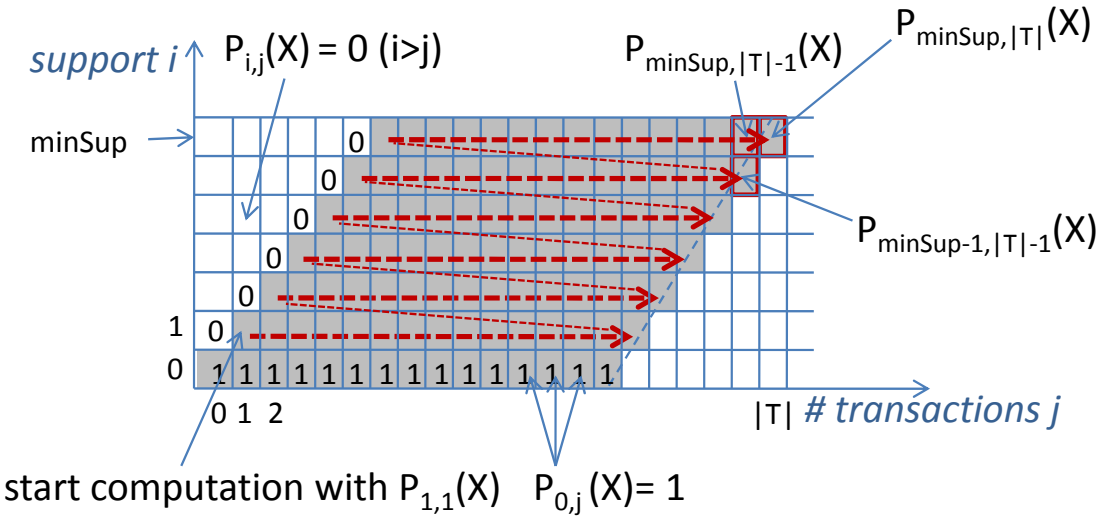


Figure 10.4: Dynamic Computation Scheme

Definition 49. *The probability that i of j transactions contain itemset X is*

$$P_{i,j}(X) = \sum_{S \subseteq T_j : |S| = i} \left(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T_j - S} (1 - P(X \subseteq t)) \right)$$

where $T_j = \{t_1, ..., t_j\} \subseteq T$ is the set of the first j transactions. Similarly, the probability that at least *i of j transactions contain itemset X is*

$$P_{\geq i,j}(X) = \sum_{S \subseteq T_j : |S| \geq i} \left(\prod_{t \in S} P(X \subseteq t) \cdot \prod_{t \in T_j - S} (1 - P(X \subseteq t)) \right)$$

Note that $P_{\geq i,|T|}(X) = P_{\geq i}(X)$, the probability that at least i transactions in the entire database contain X. The key idea in our approach is to split the problem of computing $P_{\geq i,|T|}(X)$ into smaller problems $P_{\geq i,j}(X)$, $j < |T|$. This can be achieved as follows. Given a set of j transactions $T_j = \{t_1, ..., t_j\} \subseteq T$: If we assume that transaction t_j contains itemset X, then $P_{\geq i,j}(X)$ is equal to the probability that at least $i-1$ transactions of $T_j \backslash \{t_j\}$ contain X. Otherwise, $P_{\geq i,j}(X)$ is equal to the probability that at least i transactions of $T_j \backslash \{t_j\}$ contain X. By splitting the problem in this way we can use the recursion in Lemma 30, which is an adaption of the Poison-binomial recurrence, using the paradigm of dynamic programming.

Lemma 30. $P_{\geq i,j}(X) =$

$$P_{\geq i-1,j-1}(X) \cdot P(X \subseteq t_j) + P_{\geq i,j-1}(X) \cdot (1 - P_j(X \subseteq t_j))$$

where

$$P_{\geq 0,j} = 1 \ \forall .0 \leq j \leq |T|, \ P_{\geq i,j} = 0 \ \forall .i > j$$

Proof. $P_{\geq i,j}(X) = \sum_{k=i}^{j} P_{k,j}(X) \overset{[Kollios\,et\,al]}{=}$
$\sum_{k=i}^{j} P_{k-1,j-1}(X) \cdot P(X \subseteq t_j) + \sum_{k=i}^{j} P_{k,j-1}(X) \cdot (1 - P(X \subseteq t_j)) \overset{[P_{\geq i,j}=0\ \forall .i>j]}{=} P(X \subseteq t_j) \cdot \sum_{k=i}^{j-1} P_{k-1,j-1}(X) + \cdot (1 - P(X \subseteq t_j)) \cdot \sum_{k=i}^{j-1} P_{k,j-1}(X) = P(X \subseteq t_j) \cdot P_{\geq i-1,j-1}(X) + (1 - P(X \subseteq t_j)) \cdot P_{\geq i,j-1}(X)$. □

Using this dynamic programming scheme, we can compute the probability that at least *minSup* transactions contain itemset X by calculating the cells depicted in Figure 10.4. In the matrix, each cell relates to a probability $P_{\geq i,j}$, with i marked on the x-axis, and j marked on the y-axis. Note that according to Lemma 30, in order to compute a $P_{\geq i,j}$, we require the probabilities $P_{\geq i-1,j-1}$ and $P_{\geq i,j-1}$, that is, the cell to the left and the cell to the lower left of $P_{\geq i,j}$. Knowing that $P_{\geq 0,0} = 1$ and $P_{\geq 1,0} = 0$ by definition, we can start by computing $P_{\geq 1,1}$. The probability $P_{\geq 1,j}$ can then be computed by using the previously computed $P_{\geq 1,j-1}$ for all j. $P_{\geq 1,j}$can, in turn, be used to compute $P_{\geq 2,j}$. This iteration continues until i reaches *minSup*, so that finally we obtain $P_{\geq minSup,|T|}$ – the frequentness probability (Definition 48). Note that in each line (i.e. for each i) of the matrix in Figure 10.4, j only runs up to $|T| - minSup + i$. Larger values of j are not required for the computation of $P_{minSup,|T|}$.

Lemma 31. *The computation of the frequentness probability $P_{\geq minSup}$ requires at most $O(|T| * \text{minSup}) = O(|T|)$ time and at most $O(|T|)$ space.*

Proof. Using the dynamic computation scheme as shown in Figure 10.4, the number of computations is bounded by the size of the depicted matrix. The matrix contains $|T| * minSup$ cells. Each cell requires an iteration of the dynamic computation (c.f. Corollary 30) which is performed in $O(1)$ time. Note that a matrix is used here for illustration purpose only. The computation of each probability $P_{i,j}(X)$ only requires information stored in the current line and the previous line to access the probabilities $P_{i-1,j-1}(X)$ and $P_{i,j-i}(X)$. Only these two lines (of length $|T|$) need to be preserved requiring $O(|T|)$ space. Additionally, the probabilities $P(X \subseteq t_j)$ have to be stored, resulting in a total of $O(|T|)$ space. □

Note that we can save computation time if an itemset is certain in some transactions. If a transaction $t_j \in T$ contains itemset X with a probability of zero, i.e. $P(X \subseteq t_j) = 0$, transaction t_j can be ignored for the dynamic computation because $P_{\geq i,j}(X) = P_{\geq i,j-1}(X)$ holds (Lemma 30). If $|T'|$ is less than *minSup*, then X can be pruned since, by definition, $P_{\geq minSup,T'} = 0$ if $minSup > T'$. The dynamic computation scheme can also omit transactions T_j where the item has a probability of 1, because $P_{\geq i,j}(X) = P_{\geq i-1,j-1}(X)$ due to $P(X \subseteq t_j) = 1$. Thus, if a transaction t_j contains X with a probability of 1, then t_j (i.e. the corresponding column) can be omitted if *minSup* is reduced by one, to compensate the missing transaction. The dynamic programming scheme therefore only has to consider uncertain items. We call this trick *0-1-optimization*.

10.3.2 Probabilistic Filter Strategies

To further reduce the computational cost, we introduce probabilistic filter strategies. These reduce the number of probability computations in the dynamic algorithm. Our probabilistic filter strategies exploit the following monotonicity criteria;

Monotonicity Criteria

First, if we increase the minimal support i, then the frequentness probability of an itemset decreases.

Lemma 32. $P_{\geq i,j}(X) \geq P_{\geq i+1,j}(X)$.

Proof. $P_{\geq i+1,j}(X) \overset{Definition\,48}{=} P_{\geq i,j}(X) - P_{i+1,j}(X) \leq P_{\geq i,j}(X)$ □

Intuitively, this result is obvious since the predicate "the support is at least i" implies "the support is at least $i+1$". The next criterion says that an extension of the uncertain transaction database leads to an increase of the frequentness probability of an itemset.

Lemma 33. $P_{\geq i,j}(X) \leq P_{\geq i,j+1}(X)$.

Proof. $P_{\geq i,j+1}(X) \overset{Lemma\,30}{=} P_{\geq i-1,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X) \cdot (1 - P(X \subseteq t_{j+1})) \overset{Lemma\,32}{\geq}$
$P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X) \cdot (1 - P(X \subseteq t_{j+1})) = P_{\geq i,j}(X)$ □

The intuition behind this lemma is that one more transaction can increase the support of an itemset. Putting these results together;

Lemma 34. $P_{\geq i,j}(X) \geq P_{\geq i+1,j+1}(X)$.

Proof. $P_{\geq i+1,j+1}(X) \overset{Corollary30}{=} P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i+1,j}(X)(1 - P(X \subseteq t_{j+1})) \overset{Lemma32}{\leq}$
$P_{\geq i,j}(X) \cdot P(X \subseteq t_{j+1}) + P_{\geq i,j}(X)(1 - P(X \subseteq t_{j+1})) = P_{\geq i,j}$. □

Next, we describe how these monotonicity criteria can be exploited to prune the dynamic computation.

Pruning Criterion

Lemma 34 can be used to quickly identify non-frequent itemsets. Figure 10.5 shows the dynamic programming scheme for an itemset X. Keep in mind that the goal is to compute $P_{minSup,|T|}(X)$. Lemma 34 states that the probabilities $P_{minSup-k,|T|-k}(X), 1 \leq k \leq minSup$ (highlighted in Figure 10.5), are conservative bounds of $P_{minSup,|T|}(X)$. Thus, if any of the probabilities $P_{minSup-k,|T|-k}(X)$, $1 \leq k \leq minSup$ is lower than the user specified parameter τ, then X can be pruned.

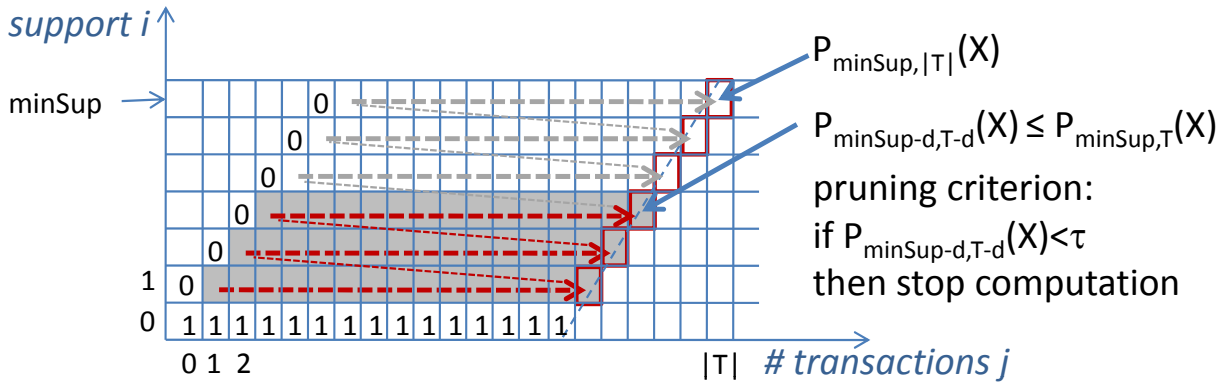


Figure 10.5: Visualization of the Pruning Criterion

10.4 Probabilistic Frequent Itemset Mining (PFIM)

We now have the techniques required to efficiently identify whether a given itemset X is a probabilistic frequent itemset (PFI). In this section, we show how to find all probabilistic frequent itemsets in an uncertain transaction database. Traditional frequent itemset mining is based on support pruning by exploiting the anti-monotonic property of support: $S(X) \leq S(Y)$ where $S(X)$ is the support of X and $Y \subseteq X$. In uncertain transaction databases however, recall that support is defined by a probability distribution and that we mine itemsets according to their frequentness probability. It turns out that the frequentness probability is anti-monotonic:

Lemma 35. $\forall Y \subseteq X : P_{\geq minSup}(X) \leq P_{\geq minSup}(Y)$. *In other words, all subsets of a probabilistic frequent itemset are also probabilistic frequent itemsets.*

Proof. $P_{\geq i}(X) = \frac{1}{|W|} \sum_{i=1}^{|W|} P(w_i) \cdot I_{S(X,w_i) \geq minSup}$, since the probability is defined over all possible worlds. Here, I_Z is an indicator variable that is 1 when $z = true$ and 0 otherwise. In other words, $P_{\geq i}(X)$ is the relative number of worlds in which $S(X) \geq minSup$ holds, where each occurrence is weighted by the probability of the world occurring. Since world w_i corresponds to a normal transaction database with no uncertainty, $S(X, w_i) \leq S(Y, w_i) \forall Y \subseteq X$ due to the anti-monotonicity of support. Therefore, $I_{S(X,w_i) \geq minSup} \leq I_{S(Y,W_i) \geq minSup} \forall i \in |W|, \forall Y \subseteq X$ and, thus, $P_{\geq i}(X) \leq P_{\geq i}(Y), \forall Y \subseteq X$. □

The above lemma shows that the frequentness probability is anti-monotonic (downward closed) in the item sets.

We can use the contra-positive of Lemma 35 to prune the search space for probabilistic frequent itemsets. That is, if an itemset Y is not a probabilistic frequent itemset, i.e. $P_{\geq minSup}(Y) < \tau$, then all itemsets $X \supseteq Y$ cannot be probabilistic frequent itemsets either. In this case, we do not need to generate further itemsets by extending X.

Our first algorithm is based on a "marriage" of traditional frequent itemset mining methods and our uncertain itemset identification algorithms. In particular, we propose a probabilistic frequent itemset mining approach based on the Apriori algorithm ([8]).

Like Apriori, our method iteratively generates the probabilistic frequent itemsets using a bottom-up strategy. Each iteration is performed in two steps, a join step for generating new candidates and a pruning step for calculating the frequentness probabilities and extracting the probabilistic frequent itemsets from the candidates. The pruned candidates are, in turn, used to generate candidates in the next iteration. Lemma 35 is exploited in the join step to limit the candidates generated and in the pruning step to remove itemsets that need not be expanded. First, we start with singleton items and compute their frequentness probability. All items having at least a frequency probability of τ are used to build all itemsets containing two items (join step). For these itemsets of size two, we compute the frequency probability, correspondingly, and filter out those having a frequency probability less than τ (pruning step). In the next iteration, we again build itemsets with three items from the remaining uncertain frequent itemsets of the previous iteration. This breadth-first strategy will be continued until all uncertain frequent itemsets are identified.

10.5 Incremental Probabilistic Frequent Itemset Mining (I-PFIM)

Our probabilistic frequent itemset mining approach allows the user to control the confidence of the results using τ. However, since the number of results depends on τ, it may prove difficult for a user to correctly specify this parameter without additional domain knowledge. Therefore, this Section shows how to efficiently solve the following problems, which do not require the specification of τ;

- *Top-k probabilistic frequent itemsets query:* return the k itemsets that have the highest frequentness probability, where k is specified by the user.
- *Incremental ranking queries:* successively return the itemsets with the highest frequentness probability one at a time.

10.5.1 Incremental Probabilistic Frequent Itemset Mining Algorithm

In our incremental algorithm (Algorithm 9), we keep an *Active Itemsets Queue (AIQ)* that is initialized with all one-item sets. The *AIQ* is sorted by frequentness probability in descending order. Without loss of generality, itemsets are represented in lexiographical order to avoid generating them more than once. In each iteration of the algorithm, i.e. each call of the *getNext()*-function, the first itemset X in the queue is removed. X is the next most probable frequent itemset because all other itemsets in the *AIQ* have a lower frequentness probability due to the order on *AIQ*, and all of X's supersets (which have not yet been generated) cannot have a higher frequentness probability due to Lemma 35. Before X is returned to the user, it is refined in a candidate generation step. In this step, we create all supersets of X obtained by adding single items x to the end of X, in such

Algorithm 9 Incremental Algorithm

//initialise
AIQ = new PriorityQueue
FOR EACH $x \in I$
  AIQ.add([x,$P_{\geq minSup}(x)$])
//return the next probabilistic frequent itemset
getNext() RETURNS X
  $X = AIQ$.removeFirst()
  FOR EACH ($x \in I \setminus X : x = lastInLexOrder(X \cup x)$)
    AIQ.add([$X \cup x$,$P_{\geq minSup}(X \cup x)$])

a way that the lexiographical order of $X \cup x$ is maintained. These are then added to the AIQ after their respective frequentness probabilities are computed (Section 10.3). The user can continue calling the *getNext()*-function until he has all required results. Note that during each call of the *getNext()*-function, the size of the AIQ increases by at most $|I|$. The maximum size of the AIQ is $2^{|I|}$, which is no worse than the space required to sort the output of a non-incremental algorithm. The runtime of each call to *getNext()* is $O(|I| \cdot |T| \cdot minSup)$.

10.5.2 Top-k Probabilistic Frequent Itemsets Query

In many applications however, relatively few top probabilistic frequent itemsets are required. For instance, in the spatial co-location application in Example 22, a user may want to know the top $k = 100$ most likely co-locations. Top-k highest frequentness probability queries can be efficiently computed by using Algorithm 9 and constraining the length of the AIQ to $k - m$, where m is the number of highest frequentness probability items already returned. Any itemsets that "fall off" the end can safely be ignored. The rational behind this approach is that for an itemset X at position p in the AIQ, $p-1$ itemsets with a higher frequentness than X exist in the AIQ by construction. Additionally, any of the m itemsets that have already been returned must have a higher frequentness probability. Consequently, our top-k algorithm contrains the size of the initial AIQ to k and reduces its size by one each time a result is reported. The algorithm terminates once the size of the AIQ reaches zero.

10.6 Experimental Evaluation

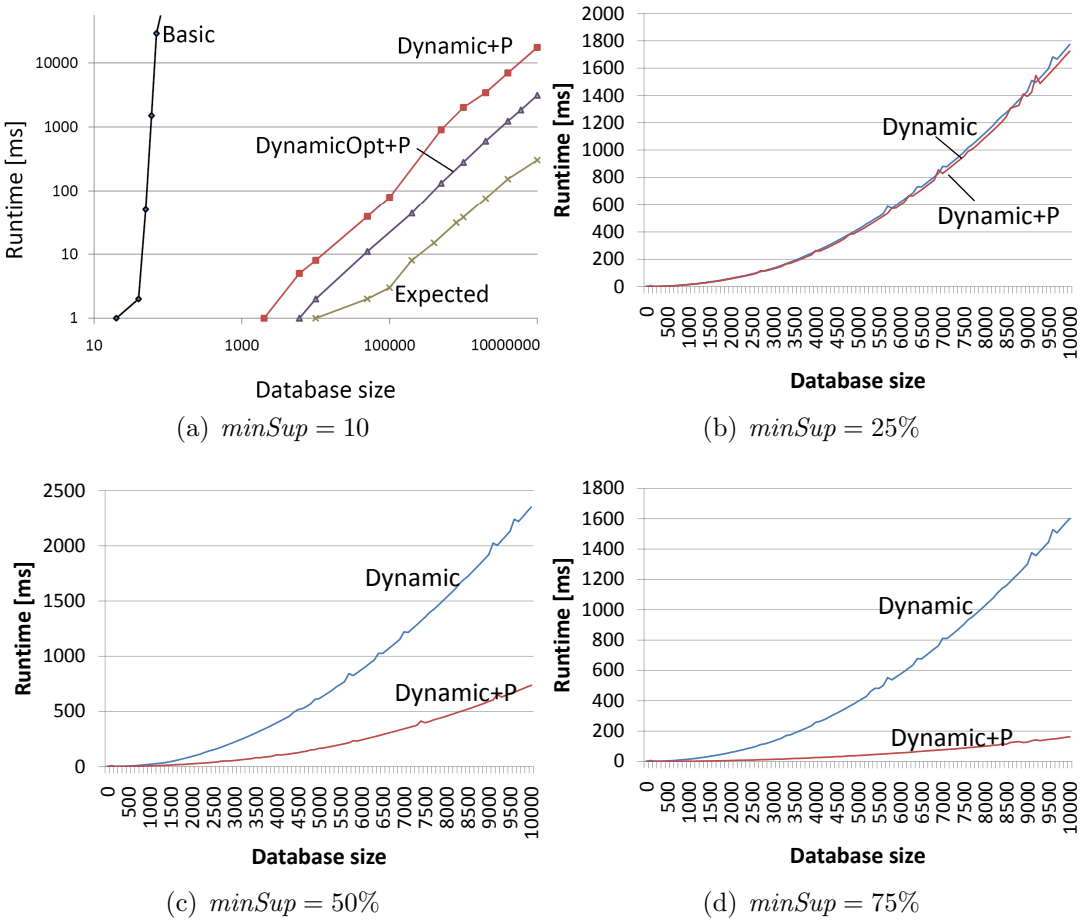
In this Section we present efficiency and efficacy experiments. First, we give efficiency results obtained utilizing the different methods of computing the probabilistic support (cf. Sections 10.2 and 10.3). Then, we discuss the performance and utility of the proposed probabilistic frequent itemset mining algorithms (cf. Sections 10.4 and 10.5). In all experiments, the runtime was measured in milliseconds (ms).

10.6.1 Evaluation of the Frequentness Probability Calculations

We evaluated our frequentness probability calculation methods on several artificial datasets with varying database sizes $|T|$ and densities. All artificial data sets have been produced by the IBM *QUEST* Market-Basket Synthetic Data Generator[1]. The number of transactions is scaled from $10k$ to $10,000k$, using $10k$ as default setting. The number of distinct items that can appear in a transaction is set to $1k$. The *density* of an item denotes the expected fraction of transactions in which an item may be present (i.e. where its existence probability is in $(0, 1]$). By default, the density is set to 0.5. The IBM data generator yields deterministic (crisp) transaction with no uncertainty involved. To add uncertainty, each item in each transaction is given a probability drawn from a uniform $[0, 1]$ distribution. The frequentness probability threshold τ of was set to 0.9 by default.

We use the following notations for our frequentness probability algorithms: **Basic**: basic probability computation (Section 10.2.2), **Dynamic**: dynamic probability computation (Section 10.3.1), **Dynamic+P**: dynamic probability computation with pruning (Section 10.3.2), **DynamicOpt**: dynamic probability computation utilizing 0-1-optimization (Section 10.3.1) and **DynamicOpt+P**: 0-1-optimized dynamic probability computation method with pruning.

[1]Original files seemingly no longer available through IBM, but mirrored at `http://www.cs.loyola.edu/~cgianneli/assoc_gen.html`



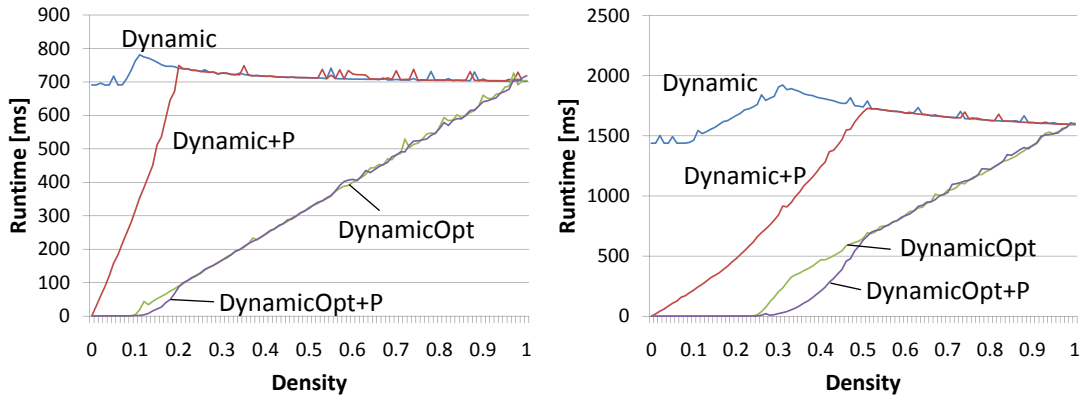
(a) $minSup = 10$ (b) $minSup = 25\%$

(c) $minSup = 50\%$ (d) $minSup = 75\%$

Figure 10.6: Runtime evaluation w.r.t. $|T|$.

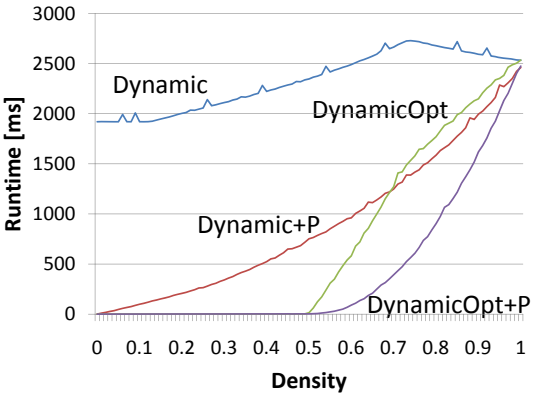
Scalability

Figure 10.6 shows the scalability of the probability calculation approaches when we vary the number of transactions, $|T|$. The runtime of the **Basic** approach increases exponentially in *minSup* as explained in Section 10.2.2, and is therefore not applicable for a $|T| > 50$ as can be seen in Figure 10.6(a). Our approaches **Dynamic+P** and **DynamicOpt+P** scale linearly as expected when using a constant *minSup* value. The 0-1-optimization has an impact on the runtime whenever there is some certainty in the database. The performance gain of our pruning strategies depends on the used *minSup* value. In Figures 10.6(b), 10.6(c) and 10.6(d) the scalability of **Dynamic** and **Dynamic+P** is shown for different *minSup* values expressed as percentages of $|T|$. It is notable that the time complexity of $O(|T| * minSup)$ becomes $O(|T|^2)$ if *minSup* is chosen relative to the database size. Also, it can be observed that the higher *minSup*, the higher the difference between **Dynamic** and **Dynamic+P**; a higher *minSup* causes the frequentness probability to fall overall, thus allowing earlier pruning.

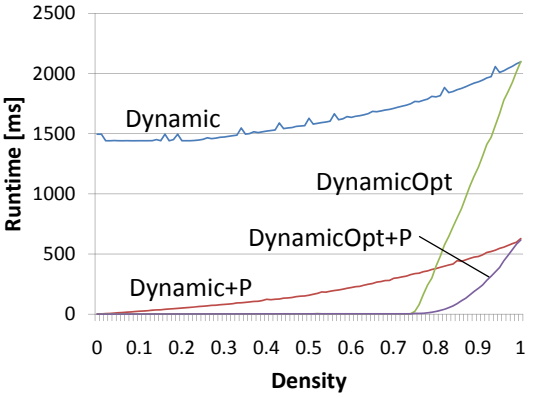


(a) $minSup = 0.1$

(b) $minSup = 0.25$



(c) $minSup = 0.5$



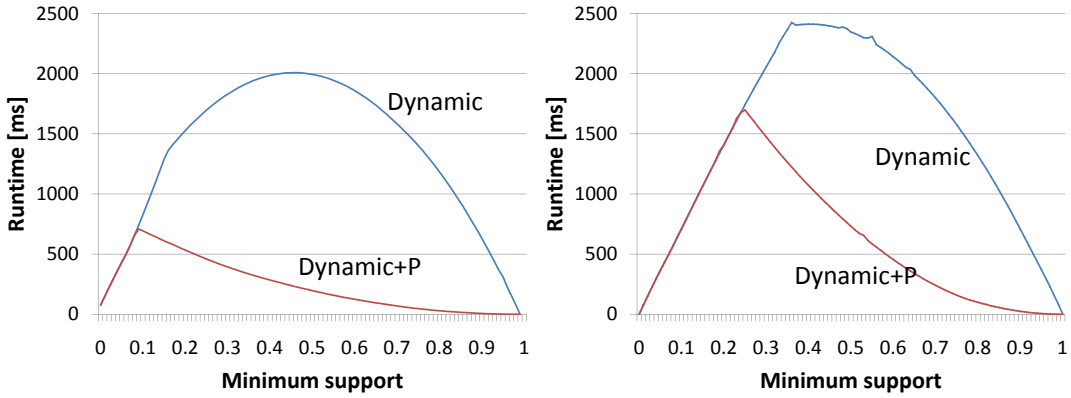
(d) $minSup = 0.75$

Figure 10.7: Runtime evaluation w.r.t. the density.

Effect of the Density

We now evaluate the effectiveness of our pruning strategy w.r.t. the density. *minSup* is important here too, so we report results for different values in Figure 10.7. The pruning works well for datasets with low density and has no effect on the runtime for higher densities. The reason is straightforward; the higher the density, the higher the probability that a given itemset is frequent and, thus, cannot be pruned. Regarding the effect of *minSup*; a larger *minSup* value decreases the probability that itemsets are frequent and therefore increases the number of computations that can be pruned. The break-even point between pruning and non-pruning in our experiments is when the density is approximately twice the *minSup* value, since, due to the method of creating our datasets, this corresponds to the expected support. At this value, all itemsets are expected to be frequent.

Overall, with reasonable parameter settings our pruning strategies achieve a significant speed-up for the identification of probabilistic frequent itemsets.

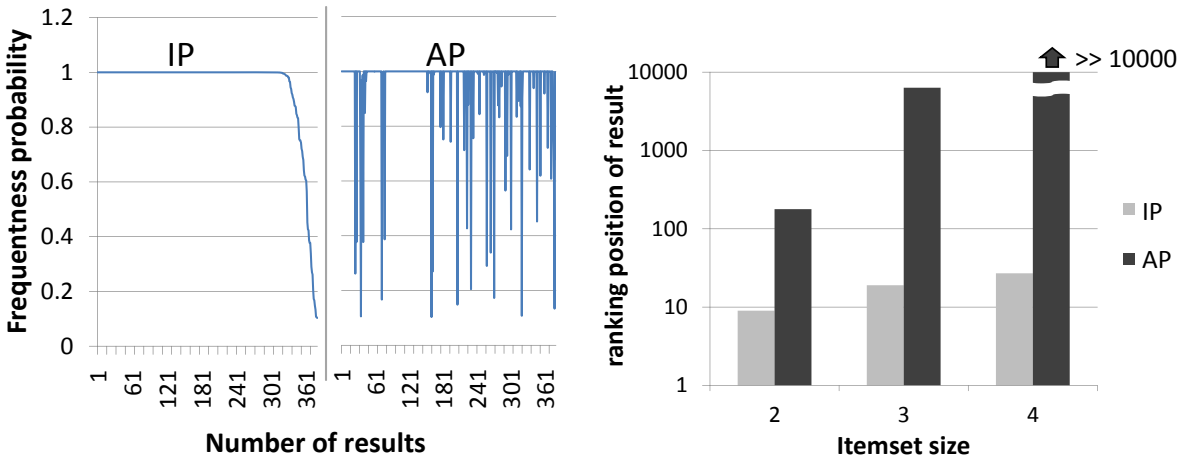


(a) Density = 0.2 (b) Density = 0.5

Figure 10.8: Runtime evaluation w.r.t. *minSup*.

Effect of *minSup*

Figure 10.8 shows the influence of *minSup* on the runtime when using different densities. The runtime of **Dynamic** directly correlates with the size of the dynamic computation matrix (see Figure 10.4). A low *minSup* value leads to few matrix rows which need to be computed, while a high *minSup* value leads to a slim row width. The total number of matrix cells to be computed is $minSup * (|T| - minSup + 1)$, with a maximum at $minSup = \frac{|T|+1}{2}$. As long as the *minSup* value is below the expected support value, the approach with pruning shows similar characteristics; in this case, almost all item(sets) are expected to be frequent. However, the speed-up due to the pruning rapidly increases for *minSup* above this break-even point.



(a) Output: AP vs. IP

(b) Effectiveness of ranking queries

Figure 10.9: Effectiveness of AP vs IP.

10.6.2 Evaluation of the Probabilistic Frequent Itemset Mining Algorithms

Experiments for the probabilistic frequent itemset mining algorithms were run on a subset of the real-world dataset *accidents*[2], denoted by ACC. It consists of 340,184 transactions and 572 items whose occurrences in transactions were randomized; with a probability of 0.5, each item appearing for certain in a transaction was assigned a value drawn from a uniform distribution in $(0, 1]$. Here we use **AP** to denote the Apriori-based and **IP** for the incremental probabilistic itemset mining algorithms.

We performed Top-k queries on the first 10,000 transactions of ACC using a $minSup = 500$ and $\tau = 0.1$. Figure 10.9(a) shows the result of **IP**. Note that the frequentness probability of the resulting itemsets is monotonically decreasing. In contrast, **AP** returns probabilistic frequent itemsets in the classic way; in descending order of their size, i.e. all itemsets of size one are returned first, etc. While both approaches return probabilistic frequent itemsets, **AP** returns an arbitrary frequentness probability order, while **IP** returns the most relevant itemsets first.

Next we performed ranking queries on the first 100,000 itemsets (Figure 10.9(b)). In this experiment, our aim was to find the m-itemset X with the highest frequency probability of all m-itemsets, where $m \in \{2, 3, 4\}$. We measured the number of itemsets returned before X. It can be seen that the speed up factor for ranking (and thus top-k queries) is several orders of magnitude and increases exponentially in the length of requested itemset length. The reason is that **AP** must return all frequent itemsets of length $m - 1$ before processing itemsets of length m, while **IP** is able to quickly rank itemsets in order of their frequentness probability, therefore leading to better quality results delivered to the user much earlier.

[2]The *accidents* dataset [77] was derived from the Frequent Itemset Mining Dataset Repository (http://fimi.cs.helsinki.fi/data/)

10.7 Conclusion

The Probabilistic Frequent Itemset Mining (PFIM) problem is to find itemsets in an uncertain transaction database that are (highly) likely to be frequent. To the best of our knowledge, this is the first work addressing this problem under possible worlds semantics. We presented a framework for efficient probabilistic frequent itemset mining. We theoretically and experimentally showed that our proposed dynamic computation technique is able to compute the exact support probability distribution of an itemset in linear time w.r.t. the number of transactions instead of the exponential runtime of a non-dynamic computation. Furthermore, we demonstrated that our probabilistic pruning strategy allows us to prune non-frequent itemsets early leading to a large performance gain. In addition, we introduced an iterative itemset mining framework which reports the most likely frequent itemsets first.

Chapter 11

Approximate Spatial Collocation Mining

This chapter presents approximation techniques for the problem of probabilistic frequent itemset mining in uncertain transaction data. Three approximation approches are presented, each based on a fundamental law of statistics.

- Supported by the Law of Large Numbers, the first approach uses expected support only, similar to [53, 54, 6].
- The Poisson Law of Small Numbers is exploited for the second approach, to approximate the probabilistic support PDF of an itemset by a Poisson distribution.
- The Central Limit Theorem is applied, allowing the third approaches to use a Normal distribution to approximate the probabilistic support PDF of an itemset.

Theoretical and experimental evaluations of all approximations are given in Section 11.2 and in Section 11.3, respectively.

11.1 Approximation of the Support PDF of an Itemset

From the last chapter, we can see that the PDF $supp(I)$ of the probabilistic support of an itemset I plays an important role in determining whether I is a probabilistic frequent itemset. However, directly computing $supp(I)$ using the Poisson binomial recurrence approach presented in Chapter 10 can be expensive due to the incurred runtime linear in the size of the database (c.f. Lemma 31). We now investigate alternative ways of computing $supp(I)$. In the following, we study some statistical properties of $supp(I)$ and show how to approximate the distribution of $supp(I)$ in a computationally efficient way by means of the expected support (cf. Section 11.1.1) and two standard probability distributions: the Poisson distribution (cf. Section 11.1.2) and the normal distribution (cf. Section 11.1.3). In Section 11.1.4 we discuss all three alternatives.

11.1.1 Approximation by Expected Support

A simple and efficient way to evaluate the frequentness of an item set in an uncertain transaction database is to use the expected support [54, 6]. The expected support converges to the exact support when increasing the number of transactions according to the "law of large numbers".

Definition 50 (Law of Large Numbers)**.** *A "law of large numbers" is one of several theorems expressing the idea that as the number of trials of a random process increases, the percentage difference between the expected and actual values goes to zero. Formally, given a sequence of independent and identically distributed random variables $X_1, ..., X_n$, the sample average $\frac{1}{n}\sum_{i=1}^{n} X_i$ converges to the expected value $\mu = \sum_{i=1}^{n} E(X_i)$ for $n \to \infty$. It can also be shown ([70]), that the law of large numbers is applicable for non-identically distributed random variables.*

For notational convenience, let p_j^I be $Pr(I \subseteq t_j)$. Since the expectation of a sum is the sum of the expectations, the expected value of X^I, denoted by μ_I, can be computed by:

$$\mu_I = \sum_{j=1}^{n} p_j^I \tag{11.1}$$

Given the expected support μ_I we can approximate the cdf $Pr_{\leq}^I(i)$ of X^I as follows:

$$Pr_{\leq}^I(i) = \begin{cases} 0 & \text{if } i < \mu_I \\ 1 & else \end{cases} \tag{11.2}$$

According to the above equation, the frequentness probability $Pr_{freq}(I)$ of itemset I is approximated by 1, if μ_I is at least *minsup* and 0 otherwise. The computation of μ_I can be efficiently done by scanning $\mathcal{DB}$ once and summing up p_j^I's for all tuples t_j in $\mathcal{DB}$.

Example 25. *As an example, consider an itemset I that only appears in two transactions with a non-zero probability. One transaction contains I with a probability of* 0.5 *and the other transaction contains I with a probability of* 0.33*. Using the techniques of Chapter 10, we can compute the exact probability mass of $supp(I)$ as:*

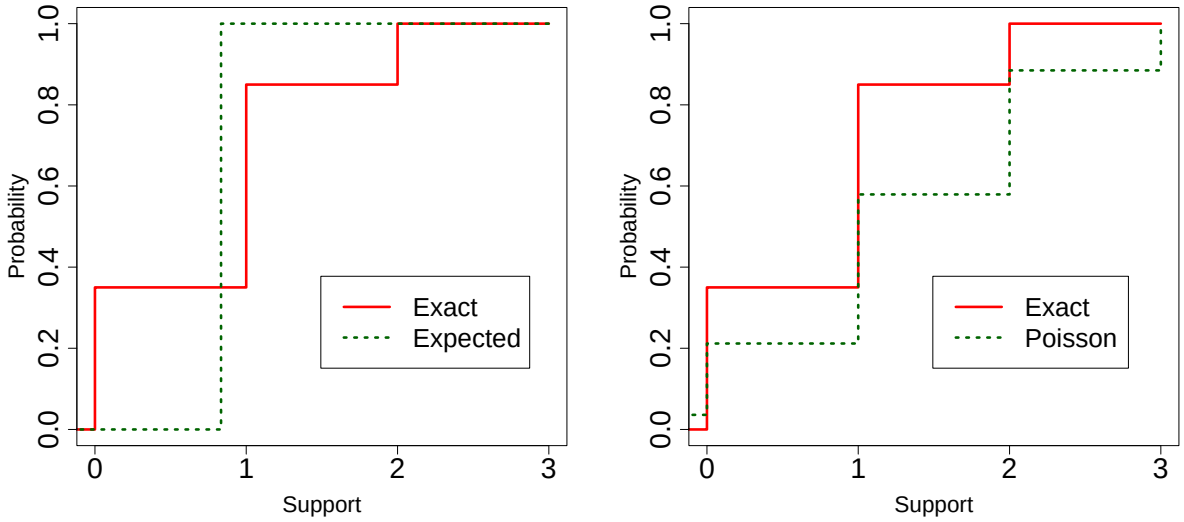
$$P(supp(I) = 0) = 0.33, P(supp(I) = 1) = 0.5, P(supp(I) = 2) = 0.17.$$

In contrast, to compute the expected support of I, we simply compute $\mu_I = 0.5+0.33 = 0.83$. The corresponding probability mass of I using expected support is

$$P(supp(I) = 0.83) = 1.$$

Both probability mass functions yield the probability density function (i.e., cumulative mass functions) depicted in Figure 11.1(a). Obviously, the expected support is only a very coarse approximation of the exact support distribution.

Important information about the distribution, e.g. the variance, is lost with this approximation. In fact, we do not know how confident the results are. In the following, we provide a more accurate approximation for the pdf of $supp(I)$.



(a) Expected approximation (b) Poisson approximation

Figure 11.1: Approximations of the support of an example itemset.

11.1.2 Poisson Distribution-Based Approximation

A Poisson binomial distribution can be well-approximated by a Poisson distribution [41] following the "Poisson law of small numbers".

Definition 51 (Poisson Law of Small Numbers)**.** *Given a sequence of independent random variables* $X_1, ..., X_n$*, having expected values* $\mu_i, 1 \leq i \leq n$ *and finite variance* ${\sigma_i}^2, 1 \leq i \leq n$*, the density of the sample average* $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ *is approximately Poisson distributed with* $\lambda_X = \frac{1}{n}\sum_{i=1}^{n}(\mu)_i$ *if* $\max\{P(X_1), ..., P(X_n)\}$ *tends to zero [89].*

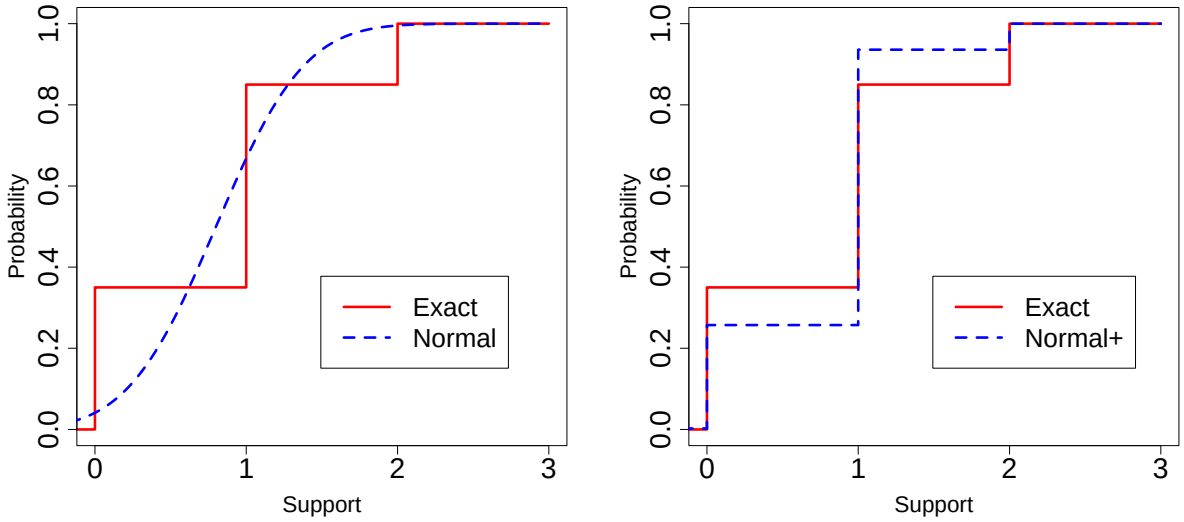
According to this law, we obtain:

$$Pr_{\geq minsup}(I) \approx 1 - Po_{\mu_I}(minsup - 1) \tag{11.3}$$

where Po_{μ_I} the cdf of the Poisson distribution with mean μ_I, i.e., $F_{\mu_I}(minsup - 1) = 1 - \frac{\Gamma(minsup, \mu_I)}{(minsup-1)!}$, where $\Gamma(minsup, \mu_I) = \int_{\mu_I}^{\infty} t^{minsup-1}e^{-t}dt$.

Example 26. *As an example, consider again the support of the itemset described in Example 25 having two existential probabilities of* 0.5 *and* 0.33*. Computing* $\mu = \lambda^I = 0.5 + 0.33$ *yields the approximated pmf depicted in Figure 11.1(b).*

To estimate $Pr_{\geq minSup}(I)$, we can first compute μ_I by scanning $\mathcal{DB}$ once as described above and evaluate $F_p(minsup - 1, \mu_I)$. Then, Equation 11.3 is used to approximate $Pr_{freq}(I)$. A theoretical analysis of the approximation quality is shown in Section 11.2.1. According to our experimental results in Section 11.3, the approximation is quite accurate.



(a) Normal distribution-based approximation (b) Approximation with continuity correction

Figure 11.2: Itemset support distribution approximated with the normal distribution.

11.1.3 Normal Distribution-Based Approximation

Provided $|\mathcal{DB}|$ is large enough which usually holds for transaction databases, $supp(I)$ converges to the normal distribution with mean μ_I and variance σ_I^2, where

$$\sigma_I^2 = Var(supp(I)) = \sum_{t_j \in \mathcal{DB}} P(I \subseteq t_j) \cdot (1 - Pr(I \subseteq t_j))$$

according to the "Central Limit Theorem".

Definition 52 (Central Limit Theorem)**.** *Given a sequence of independent random variables $X_1, ..., X_n$, having expected values $\mu_i, 1 \leq i \leq n$ and finite variance ${\sigma_i}^2, 1 \leq i \leq n$, the density of the sample average $X = \frac{1}{n}\sum_{i=1}^{n} X_i$ is approximately normal distributed with $\mu_X = \frac{1}{n}\sum_{i=1}^{n} \mu_i$ and ${\sigma^2}_X = \frac{1}{\sqrt{n}}\sum_{i=1}^{n} {\sigma^2}_i$.*

Lemma 36. *The support probability distribution of an item set I is approximated by the normal distribution with mean μ_I and variance σ_I^2 as defined above. Therefore,*

$$P_{\geq minsup}(I) \approx 1 - No_{\mu_i,\sigma_I^2}(minsup - 1) \tag{11.4}$$

where No_{μ_i,σ_I^2} is the cdf of the normal distribution with mean μ_I and variance σ_I^2, i.e.,

$$No_{\mu_i,\sigma_I^2}(minsup - 1) = \frac{1}{\sigma_I\sqrt{2\pi}} \int_{-\infty}^{minsup-1} e^{-\frac{(x-\mu_I)^2}{2\sigma_I^2}} \tag{11.5}$$

Computing $\mu = 0.5 + 0.33$ and $sigma^2 = 0.5 \cdot 0.5 + 0.33 \cdot 0.67 = 0.471$ yields the approximated pmf depicted in Figure 11.2(a). The continuity correction which is achieved by running the integral up to $minsup - 0.5$ instead of $minsup - 1$ is an important and common method to compensate the fact that $supp(I)$ is a discrete distribution approximated by a continuous normal distribution. The effect of the continuity correction is shown in Figure 11.2(b).

The estimation of $P_{\geq}(I)$ can be done by first computing μ_I by scanning $\mathcal{DB}$ once, summing up p_j^I's for all tuples t_j in $\mathcal{DB}$. During the same scan, the variance $\sigma_I^2 = Var(I)$ can be computed by exploiting independence of all items to use the Bienaym formula

$$Var(\sum_{i=1} nX_i) = \sum_{i=1} nVar(X_i).$$

Now Equation 11.4 is used to approximate $P_{\leq}(I)$.

While the above method still requires a full scan of the database to evaluate one frequent itemset candidate, threshold- and rank-based PFIs can be found more efficiently.

11.1.4 Discussion

In this section, we have described three models to approximate the Poisson binomial distribution. Now, we will discuss the advantages and disadvantages of each model theoretically.

Each of the approximation models is based on a fundamental statistics theorem. In particular, the *Expected* approach, exploits the *Law of Large Numbers*[158], the *Normal Approximation* approach exploits the *Central Limit Theorem*[69] and the *Poisson Approximation* approach exploits the *Poisson Law of Small Numbers*[152].

Approximation Based on Expected Support

In consideration of the "Law of large numbers", the *Expected* approach requires a large n, i.e. a large number of transactions, where the respective item set is contained with a probability greater than zero. A thorough evaluation of this parameter can be found in the experiments.

Normal Distribution-Based Approximation

The rule of thumb for the central limit theorem is, is that for $n \geq 30$, i.e. for at least 30 transactions containing the respective item set with a probability larger than zero, the normal approximation yields good results. This rule of thumb however, depends on certain circumstances, namely, the probabilities $P(X_i = 1)$ should be close to 0.5. In our experiments, we will evaluate, for what settings (e.g. databases size, item set probabilities) the normal approximation yields good results.

Poisson Distribution-Based Approximation

In consideration of the *Poisson Law of Small Numbers*, the Poisson approximation theoretically yields good results, if *all* probabilities $P(X_i)$ are small. This seems to be a harsh assumption, since it forbids any probabilities of one, which are common in real data sets. However, it can be argued, that for large item sets, the probability may always become small in some applications. In the experiments, we will show how small $\max\{P(X_1), ..., P(X_n)\}$ is required to be, in order to achieve good approximations, and how "a few" large probabilities impact the approximation quality. In addition, our experiments aim to give an intuition, in what setting which approximation should be used to achieve the best approximation results.

Computational Complexity

Each approximation technique requires to compute the expected support $E(X) = \mu = \lambda = \sum_{i=1}^{n} P(X_i)$, which requires a full scan of the database requiring $O(n)$ time and $O(1)$ space. The normal approximation additionally requires to compute the sum of variances, which has the same complexity. This is all that has to be done to compute the parameters of all three approximations. After that, the Expected approach only requires to compare $E(X)$ with $MinSupp$, at a cost of $O(1)$ time. The normal approximation approach in contrast requires, to compute the probability that $X > MinSupp$, which requires numeric integration, since the normal distribution it has no closed-form expression. However, there exist very efficient techniques (such as the Abromowitz and Stegun approximation [2]) to quickly evaluate the normal distribution. Regardless, this evaluation is independent of the database size and also requires constant time. The same rationale applies for the Poisson approximation, which does also not have a closed-form solution, but for which there exist manifold fast approximation techniques. In summary, each of the approximation techniques has a total runtime complexity of $O(n + C_i)$ and a space complexity of $O(1)$. The constant C_i depends on the approximation technique. In the experiments, we will see, that the impact of C_i can be neglected in runtime experiments. In summary, each of the proposed approximation techniques runs in O(n) time. These are more scalable methods compared to the solution presented in Chapter 10.

11.2 Theoretical Bounds on the Approximation Quality

This section presents theoretical results on the quality of the Poisson and of the Normal approximation. An empiric evaluation can be found in Section 11.3

11.2.1 Quality of the Poisson Approximation

In Section 11.1.2, we use the Poisson distribution to approximate the Poisson binomial distribution. Here, we summarize the results of this approximation quality, discussed in [174].

Let $X_1, X_2, ..., X_n$ be a set of Poisson trials such that $Pr(X_j = 1) = p_j$ and $X = \sum_{j=1}^{n} X_j$. Then, X follows a Poisson binomial distribution. Suppose $\mu = E[X] = \sum_{j=1}^{n} p_j$. The probability of $X = i$ and $X \leq i$ can be approximated by the probability distribution function (*pdf*) and the cumulative distribution function (*cdf*) of the Poisson distribution,

$$Pr(X = i) \approx f(i, \mu) = \frac{\mu^i}{i!} e^{-\mu}$$
$$Pr(X \leq i) \approx F(i, \mu) = \frac{\Gamma(i+1, \mu)}{i!}$$

[174] gives an upper bound of the error of the approximation:

$$|Pr(X \leq i) - F(i, \mu)| \leq (\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 \tag{11.6}$$

for $i = 0, 1, 2, ..., n$ where $\mu^{-1} \wedge 1 = min(\mu^{-1}, 1)$.

Now, we want to compute a bound on expression on the right hand side. Since $\mu = \sum_{j=1}^{n} p_j$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = min(\frac{1}{\sum_{j=1}^{n} p_j}, 1) \sum_{j=1}^{n} p_j^2$$

Obviously the above expression is greater than or equal to 0.
When $0 \leq \sum_{j=1}^{n} p_j \leq 1$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = \sum_{j=1}^{n} p_j^2 \leq \sum_{j=1}^{n} p_j \leq 1$$

When $\sum_{j=1}^{n} p_j > 1$,

$$(\mu^{-1} \wedge 1) \sum_{j=1}^{n} p_j^2 = \frac{\sum_{j=1}^{n} p_j^2}{\sum_{j=1}^{n} p_j} \leq \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n} p_j} = 1$$

So, in either case:

$$0 < (\mu^{-1} \wedge 1) \sum_{i=1}^{n} p_i^2 \leq 1 \tag{11.7}$$

The upper bound of the error is very small. As also verified by our experiments, the Poisson binomial distribution can be approximated well.

11.2.2 Quality of the Normal Approximation

In Section 11.1.3 we use a normal distribution to approximate a Poisson binomial distribution. Here, we will summarize the current state of research in theoretical quality of this approximation. Therefore, let $X_1, ..., X_n$ independent Poisson trials with respective probabilities $p_1, ..., p_n$ and let X denote random variable corresponding to the average $\frac{1}{n}\sum_{i=1}^{n} X_i$ of these random variables. Also, let Y denote the CDF of $\frac{X \cdot \sqrt{(n)}}{\sigma}$, i.e. the normalized CDF of X. In [67], the author was able to prove that the maximum error between Y and the cdf Φ of the standard normal distribution is bounded as follows:

$$\sup_x |Y - \Phi| \leq C\Psi,$$

where $\Psi = (\sum_{i=1}^{n} \sigma_i^2)^{\frac{-3}{2}} \cdot \sum_{i=1}^{n} \rho_i$, where σ_i^2 is the variance of X_i and ρ_i is the third moment of X_i. C is constant that was successively lowered from the original estimate of 7.59 ([67]) to the current best estimate of 0.5600 by [169].

11.3 Experimental Results

We now present the experimental results, again based on the same datasets that have been used in the experimental Section 10.6 of the previous Chapter 10. The first one, called *accidents*, is taken from the Frequent Itemset Mining (FIMI) Dataset Repository[1]. This dataset is obtained from the National Institute of Statistics (NIS) for the region of Flanders (Belgium), for the period of 1991–2000. The data are obtained from the 'Belgian Analysis Form for Traffic Accidents', which are filled out by a police officer for each traffic accident occurring on a public road in Belgium. The dataset contains 340,184 accident records, with a total of 572 attribute values. On average, each record has 45 attributes.

We use the first 10k tuples and the first 20 attributes as our default dataset. The default value of *minsup* is 20% of the database size n.

The second dataset, called *T10I4D100k*, is again produced by the IBM data generator used in Section 10.6. The dataset has a size n of 100k transactions. On average, each transaction has 10 items, and a frequent itemset has four items. Since this dataset is relatively sparse, we set *minsup* to 1% of n.

For both datasets, we consider both attribute and tuple uncertainty models. For attribute uncertainty, the existential probability of each attribute is drawn from a Gaussian distribution with mean 0.5 and standard deviation 0.125. This same distribution is also used to characterize the existential probability of each tuple, for the tuple uncertainty model. The default values of *minprob* and k are 0.4 and 10, respectively. In the results presented, *minsup* is shown as a percentage of the dataset size n. Notice that when the values of *minsup* or *minprob* are large, no PFIs can be returned; we do not show the results for these values. Our experiments were carried out on the Windows XP operating system,

[1] `http://fimi.cs.helsinki.fi/`

on a work station with a 2.66 GHz Intel Core 2 Duo processor and 2GB memory. The programs were written in R.

We now compare the performance of five threshold-based PFI mining algorithms mentioned in this chapter: (1) `DP`, the Apriori-based algorithm used in Chapter 10; (2) `Expected`, the modified Apriori algorithm that uses the expected support only [54]; (3) `Poisson`, the modified Apriori algorithm that uses the Poisson approximation to estimate the support of an itemset; (4) `Normal`, the modified Apriori algorithm that uses the normal approximation.

minsup/*n*	0.1	0.2	0.3	0.4	0.5
Recall	0.99	0.98	1	1	1
Precision	1	1	1	1	1
Recall & *Precision* vs. *minsup*					
minprob	0.1	0.3	0.5	0.7	0.9
Recall	0.975	0.975	1	1	1
Precision	1	1	1	0.975	0.941
Recall & *Precision* vs. *minprob*					
n	1k	5k	10k	50k	100k
Recall	0.937	0.986	0.983	1	1
Precision	0.969	1	0.992	1	1
Recall & *Precision* vs. n					

(a) `Expected` approach.

minsup/*n*	0.1	0.2	0.3	0.4	0.5
Recall	1	1	1	1	1
Precision	1	1	1	1	1
Recall & *Precision* vs. *minsup*					
minprob	0.1	0.3	0.5	0.7	0.9
Recall	1	1	1	1	1
Precision	1	1	1	1	1
Recall & *Precision* vs. *minprob*					
n	1k	5k	10k	50k	100k
Recall	1	1	1	1	1
Precision	1	1	1	1	1
Recall & *Precision* vs. n					

(b) `Normal` approach.

minsup/*n*	0.1	0.2	0.3	0.4	0.5
Recall	1	1	1	1	1
Precision	1	0.992	1	1	1
Recall & *Precision* vs. *minsup*					
minprob	0.1	0.3	0.5	0.7	0.9
Recall	1	1	1	0.983	0.985
Precision	0.986	1	0.985	1	1
Recall & *Precision* vs. *minprob*					
n	1k	5k	10k	50k	100k
Recall	1	1	1	1	1
Precision	0.989	1	0.992	1	1
Recall & *Precision* vs. n					

(c) `Poisson` approach.

Table 11.1: Recall and Precision of the approximations.

11.3.1 Accuracy

Since the model-based approaches `Expected`, `Poisson` and `Normal` each approximate the exact pdf $supp(I)$ of an itemset I, we first examine their respective accuracy with respect to `DP`, which yields PFIs based on exact frequentness probabilities. Here, we use the standard *recall* and *precision* measures [40], which quantify the number of negatives and false positives. Specifically, let `MB`$\in$ {`Expected`, `Poisson`, `Normal`} be one of the model-based approximation approaches and let F_{DP} be the set of PFIs generated by `DP`, and F_{MB} be the set of PFIs produced by `MB`. Then, the recall and the precision of `MB`, relative to `DP`, can be defined as follows:

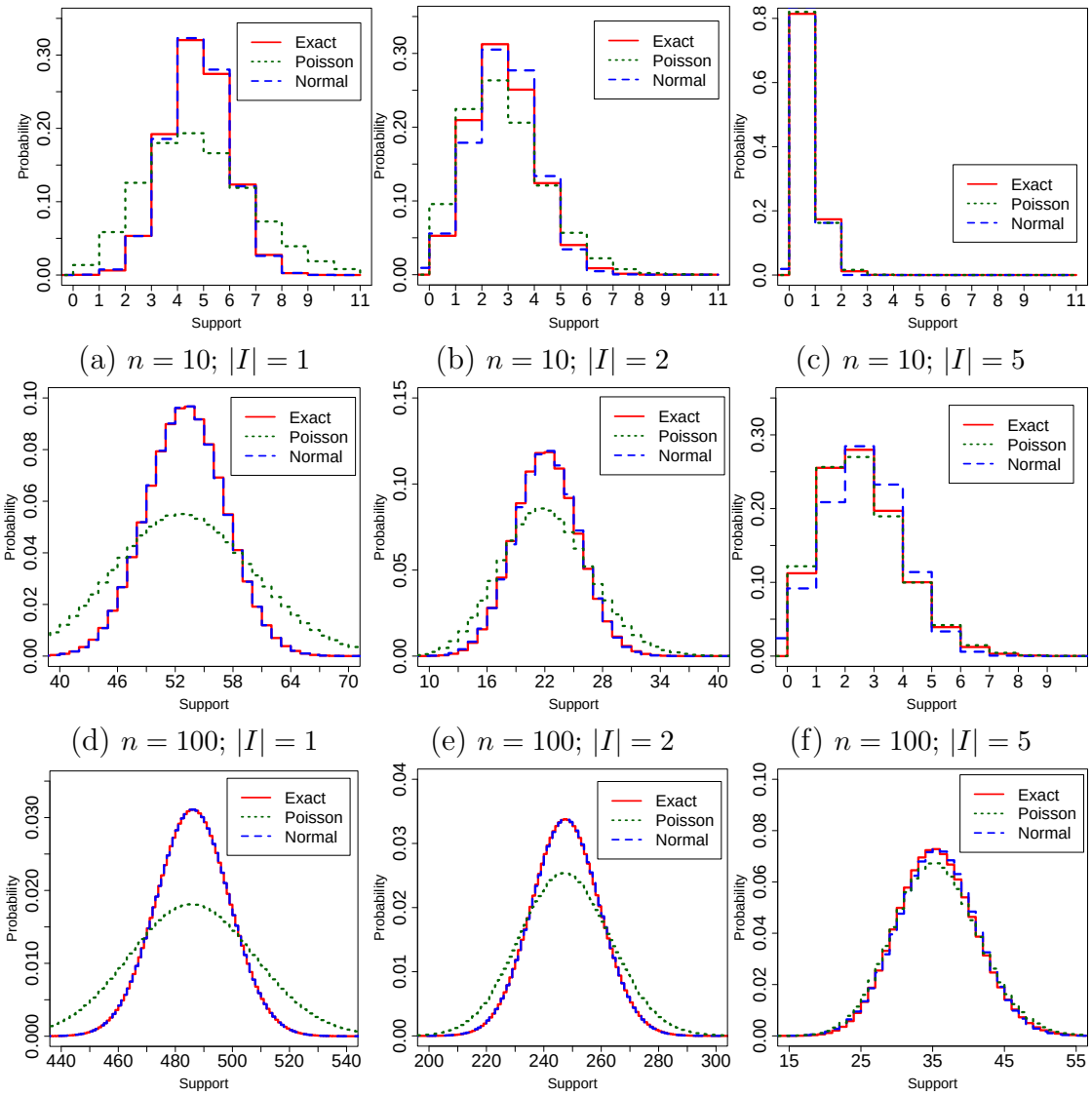
$$recall = \frac{|F_{DP} \cap F_{MB}|}{|F_{DP}|} \tag{11.8}$$

$$precision = \frac{|F_{DP} \cap F_{MB}|}{|F_{MB}|} \tag{11.9}$$

In these formulas, both recall and precision have values between 0 and 1. Also, a higher value reflects a better accuracy.

Tables 11.1(a) to (c) show the recall and the precision of the `MB` approaches, for a wide range of *minsup*, n and *minprob* values. As we can see, the precision and recall values are generally very high. Hence, the PFIs returned by the `MB` approaches are highly similar to those returned by `DP`. In particular, we see that the `Expected` approach generally yields the worst results, having precision and recall values of less than 95% in some setting. The `Poisson` approach performs significantly better in these experiments. Yet in some settings, the `Poisson` approach reports false hits, while in other settings, it performs false dismissals. The `Normal` approach is most notable in this experiment. In this set of experiments, the `Normal` approach never had a false dismissal, nor did it report a single false hit. This observation also remained true for further experiments in this line, which are not depicted here. In the next set of experiments, we will experimentally investigate the effectivity of the `MB` approach.

Figure 11.3 shows the exact probability mass functions (pmfs), as well as the pmfs approximated by the `Poisson` and the `Normal` approach, for a variety of settings. In particular, we scaled both the data set size and the size of the item sets whose pmf is to be approximated. Since, in this setting, the probabilities of individual items are uniformly sampled in the $[0, 1]$ interval, and since due to the assumption of item independence, it holds that $P(I) = \prod_{i \in I} P(i)$, a large item set I implies smaller existence probabilities. In Figure 11.3(a), it can be seen that for single item sets (i.e. large probability values), the pmf acquired by the Poisson approximation is too shallow - that is, for support values close to μ_I the exact pmf is underestimated, while for support values far from μ_I, the pmf is overestimated. In Figures 11.3(d) and 11.3(g) it can be observed, that this situation does not improve for the Poisson approximation. However, this shortcoming of the Poisson approximation can be explained: Since the Poisson approximation does only have one parameter μ_I, but no variance parameter ${\sigma^2}_I$, it is not able to differ between a set of five transactions with occurrence probabilities of 1, and five million transactions with



(a) $n = 10$; $|I| = 1$ (b) $n = 10$; $|I| = 2$ (c) $n = 10$; $|I| = 5$

(d) $n = 100$; $|I| = 1$ (e) $n = 100$; $|I| = 2$ (f) $n = 100$; $|I| = 5$

(g) $n = 1000$; $|I| = 1$ (h) $n = 1000$; $|I| = 2$ (i) $n = 1000$; $|I| = 5$

Figure 11.3: Illustration of the approximation quality of `Normal` and `Poisson` for various settings.

occurrence probabilities of 10^{-6}, since in both scenarios it holds that $\mu_I = 5 \cdot 1 = 5 \cdot 10^6 \cdot 10^{-6} = 5$. Clearly, the variance is much greater in the later case, and so are the tails of the corresponding exact pmf. Since the Poisson distribution is the distribution of the low probabilities, the Poisson assumes the later case, i.e. a case of very many transactions, each having a very low probability. In contrast, the normal distribution is able to adjust to these different situations by adjusting its ${\sigma^2}_I$ parameter accordingly. Figures 11.3(b), 11.3(e) and 11.3(h) show the same experiments for two-item sets, i.e. for lower probabilities $I \subseteq t_i$. It can be observed that with lower probabilities, the error of the Poisson approximation decreases, while (as we will see later, in Table 11.2) the quality of the Normal approximation decreases. Finally, for item sets of size 5, the Poisson approximation comes very close to the exact pmf.

		# itemsets		
n	Model	1	2	5
10	Normal	0.020	0.078	0.180
	Poisson	0.526	0.283	0.077
100	Normal	0.0003	0.0020	0.0134
	Poisson	0.0515	0.0283	0.0052
1000	Normal	2.39E-6	6.12E-6	0.0004
	Poisson	5.24E-3	2.85E-3	0.0007

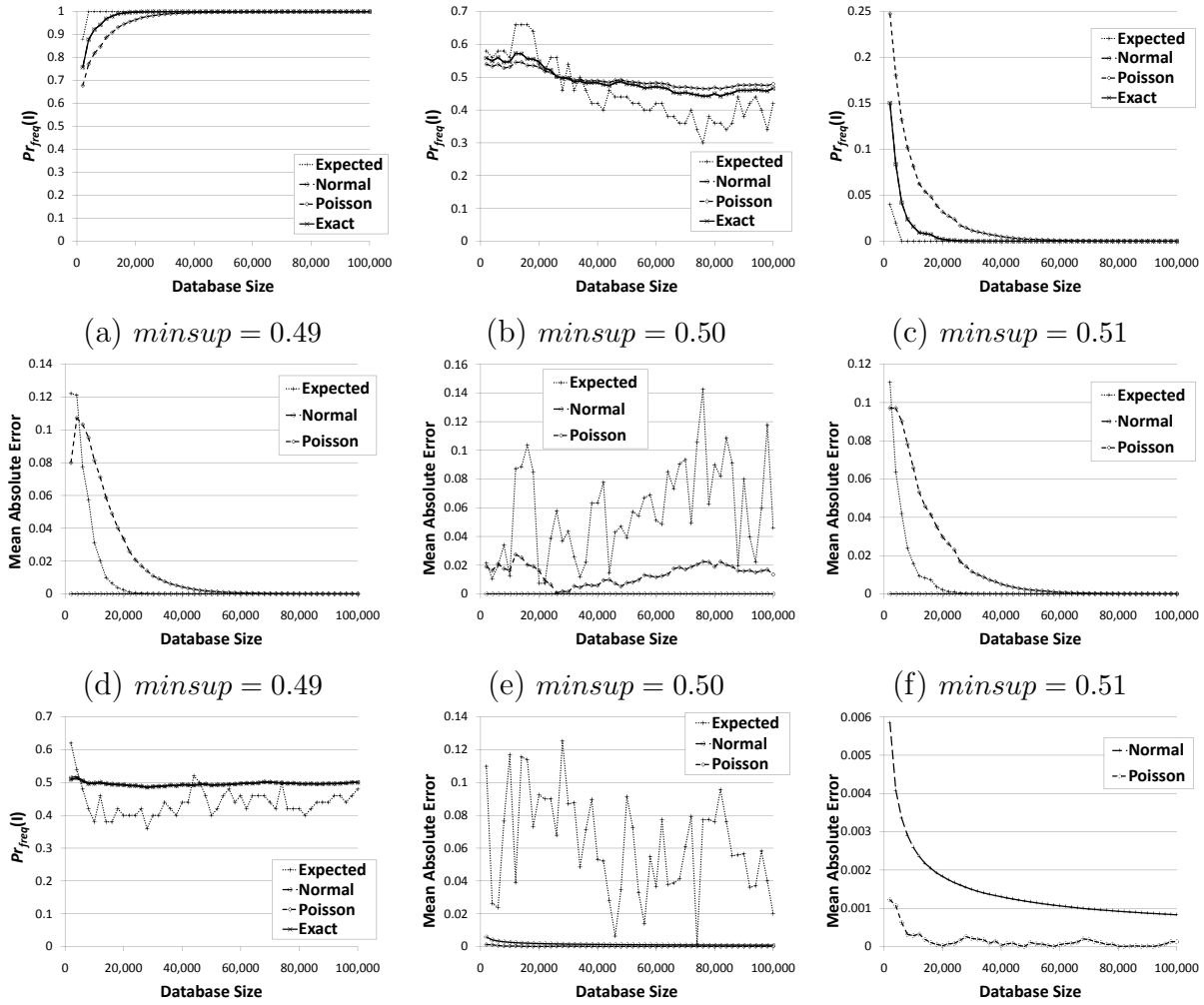
Table 11.2: Approximation Quality

To gain a better intuition of the approximation errors, we have repeated each of the experiments shown in Figure 11.3 one hundred times, and measured the total approximation error. That is, we have measured for each approximation `DP`$\in\{$`Normal`, `Poisson`$\}$ the distance to the exact pmf, computed by the `DP` approach:

$$D(MP, DP) = \sum_{i=0}^{n} |MP_{pmf} - DP_{pmf}|.$$

The results of this experiment are shown in Table 11.2. Clearly, the approximation quality of the Normal approach decreases when the size of the item sets increases (i.e. the probabilities become smaller), while the Poisson approach actually improves. An increase of the database size, is beneficial for both approximation approaches.

The impact of increasing the database size on the individual approximation models has been investigated further in Figure 11.4. In this experiment, we use a syntethic item set where each item is given a random probability uniformly distributed in $[0, 1]$. In Figures 11.4(a)-(c), the average frequentness probability $Pr_{freq}(I)$ that item I is frequent is depicted for all one-item sets I. Since all probabilities are uniformly $[0, 1]$ distributed, it is clear that μ_I is about $\frac{n}{2}$. Thus, for minsup=0.49, the probability that an item is frequent increases in the database size, for minsup=0.5, about half of the items are frequent, and for minsup=0.51, the number of frequent items decreases in the database size. First, it can be observed that the Normal approximation is nearly perfect, since no error between



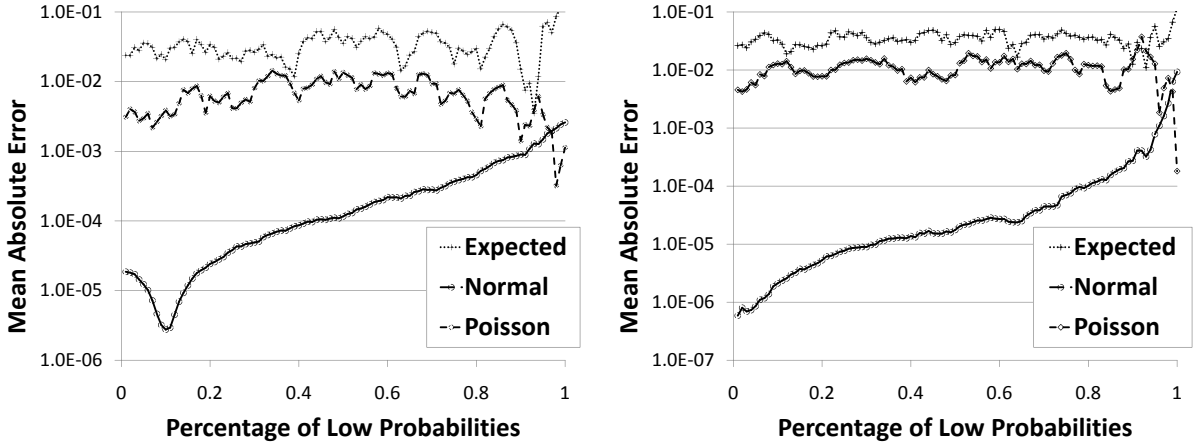
(a) $minsup = 0.49$ (b) $minsup = 0.50$ (c) $minsup = 0.51$

(d) $minsup = 0.49$ (e) $minsup = 0.50$ (f) $minsup = 0.51$

(g) p-value, $minsup = 0.50$ (h) error, $minsup = 0.50$ (i) error, $minsup = 0.50$

Figure 11.4: Accuracy of model-based algorithms vs. n.

the exact $Pr_{freq}(I)$ and it's Normal approximated value can be observed visually. This is also confirmed by Figures 11.4(d)-(f), which show the respective error, i.e. the differences between the exact frequentness probability and the approximated frequentness probabilities. In contrast, the Poisson approximation does show a significant error for smaller databases. For minsup=0.49 and minsup=0.51, the Poisson approximation is also able to yield very good approximation for database sizes larger than 50.000 while the expected support yields good results for even smaller databases. The reason is that in this case, all the exact frequentness probabilities quickly converge to 1 (0), which can easily be guessed by the expected approach. However, the interesting case is the case where minsup=0.5, since in this case, the items are expected to have an average frequentness probability of 0.5. However, in this case, the expected approach is forced to guess, since it may only take the values 0 and 1. This explains the large error of the expected approach, which does



(a) $lowP <= 0.10$ (b) $lowP <= 0.01$

Figure 11.5: Accuracy of the model-based algorithms vs. fraction of low probability values.

not decrease in the database size. The Poisson approach performs significantly better than the expected approach, but still the error is relatively high, which matches our previous experiments for large probabilities. To achieve a better setting for the Poisson approximation, we changed the interval from which the item probabilities are sampled, from $[0, 1]$ to $[0, 0.1]$ and repeated the experiment for minsup=0.5. The results are depicted in Figures 11.4(g)-(h). It can be observed that in this setting, the Poisson approximation achieves extremely good results, even slightly outperforming the Normal approximation. In Figure 11.4(i), the expected approach is omitted, to give a better view on the difference of the Normal and the Poisson error.

In the previous setting, we have evaluated the performance of the model-based approximation approaches in the case where all items have the same occurrence probabilities. We have seen that for small probabilities, the Poisson approximation works well. Next we will evaluate, how many probability values are allowed to be large, in order for the Poisson approximation to still yield good results. In the following experiment, we introduce a new parameter $lowP$, which denotes the fraction of the item probabilities, which are chosen from a smaller interval, such as the interval $[0, 0.01]$, while the remaining $1 - lowP$ fraction of the items has their probabilities chosen from the $[0, 1]$ interval. The result of the experiment evaluating the impact of $lowP$ is given in Figure 11.5. It can be seen that the mean absolute approximation error of the Normal approximation increases as the fraction $lowP$ of small probabilities increases. In contrast, the approximation error of the Poisson approximation slightly decreases. However, only for $lowP > 0.98$, the Poisson approximation begins to outperform the Normal approximation. Thus, if the data set only contains a few probability values that are not small, then the Normal approximation outperforms the Poisson approximation.

While in the previous experiment, the approximation quality has been measured for individual item sets, we now compare the effectivity of the model-based approaches for the complete Apriori-based algorithm.

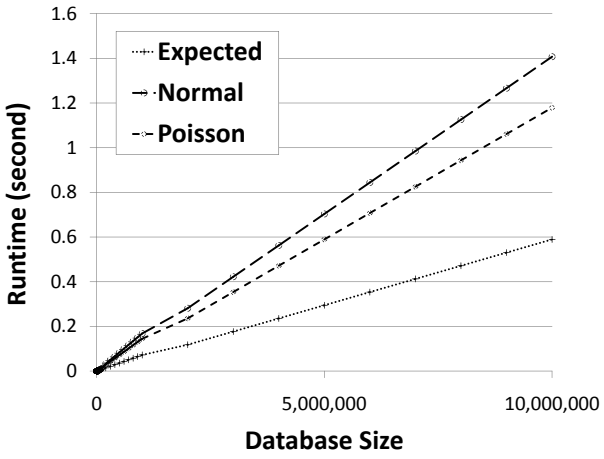
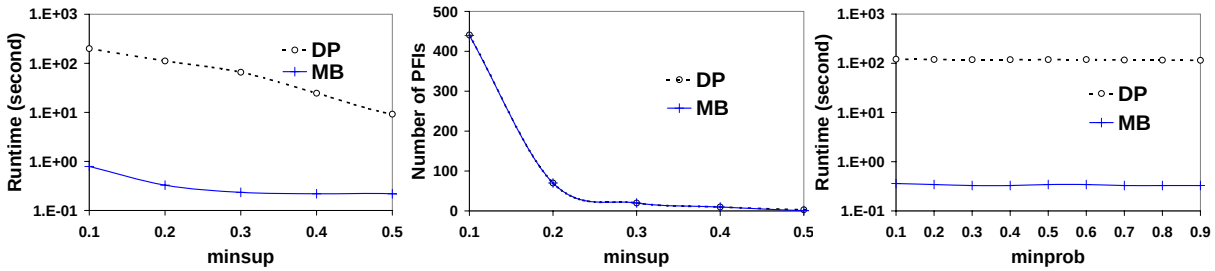


Figure 11.6: Performance comparison of model-based approaches.

11.3.2 Efficiency

Next, we compare the performance (in log scale) of the model-based approaches (MB) and the Poisson binomial recurrence based DP. First, we will compare the runtime of the three (MB)s to each other. The result is shown in Figure 11.6. It can be seen that the approach using expected support, since it only has to perform a single value comparison ($\mu_I \geq minsup$), is faster than the other model-based approaches by a factor of about two. The normal and the Poisson approximation take about the same time to compute. While in our Java experiments (shown here), the Poisson approximation slightly outperforms the normal approximation, our experiments in R (not depicted here) show the opposite situation. In summary, we can say that Poisson and normal approximation take about the same time to evaluate, except for some possibly implementation specific differences.

In the following runtime experiments, we will for simplicity only show one graph for the model-based (MB) approaches, which corresponds to the runtime of the Normal approximation. For the runtime of the expected approach, simply divide the result by two due to the observations made in Figure 11.6.



(a) Runtime vs. *minsup* (b) $|PFI|$ vs. *minsup* (c) Runtime vs. *minprob*

Figure 11.7: Threshold-based PFI Mining: Efficiency of model-based algorithm MB vs. dynamic programming DP.

Figure 11.7(a). Observe that MB is about two orders of magnitude faster than DP, over a wide range of *minsup*. This is because MB does not compute exact frequentness probabilities as DP does; instead, MB only computes the μ_I values, which can be obtained faster. We also notice that the running times of both algorithms decrease with a higher *minsup*. This is explained by Figure 11.7(b), which shows that the number of PFIs generated by the two algorithms, $|PFI|$, decreases as *minsup* increases. Thus, the time required to compute the frequentness probabilities of these itemsets decreases. We can also see that $|PFI|$ is almost the same for the two algorithms, reflecting that the results returned by MB closely resemble those of DP.

Figure 11.7(c) examines the performance of MB and DP (in log scale) over different *minprob* values. Their execution times drop by about 6% when *minprob* changes from 0.1 to 0.9. We see that MB is faster than DP. For instance, at $minprob = 0.5$, MB needs 0.3 seconds, while DP requires 118 seconds, delivering an almost 400-fold performance improvement.

11.4 Conclusions

In this chapter, model-based approaches to discover Probabilistic Frequent Itemsets (PFIs) from uncertain databases have been proposed. The proposed methods represent the probability mass function of a probabilistic frequent item using probability models, in order to find PFIs quickly. These methods can efficiently extract threshold- and rank-based PFIs. They also support attribute and tuple uncertainty models, which are two common data models. We develop new approximation methods to evaluate frequentness probabilities efficiently. As shown theoretically and experimentally, our algorithms are more efficient and scalable than existing ones. They are also highly accurate, although, some models require certain properties of the data set to be satisfied in order to achieve very high accuracy. We have theoretically and experimentally compared our proposed model-based approaches and shown properties of the data which are required for each model to perform well. To conclude, a summary of the pros and cons of each model is as follows:

11.4.1 Expected Support:

Easy to implement and efficiently to compute, since the total runtime is about twice as fast as the other approaches, and pruning can be performed. However, this approach lacks effectivity, since this model requires a very large number of transactions containing an item set I with a probability greater than zero, in order to achieve acceptable results. Except for very small item sets, which are usually not interesting because they are trivially frequent, this requirement is hardly satisfied. Also, the parameter *minprob* cannot be integrated into this approach, so that the level of significance of frequent items cannot be determined.

11.4.2 Poisson Approximation:

Easy to implement and efficiently to compute, since pruning can be performed. However, this model requires that almost all transactions containing I have a very low probability, in order to obtain good approximation results. This requirement is hardly ever met on real data, since such data sets generally contain some transaction containing I with a probability of 1. However, this model is not robust since a few larger probabilities will significantly lower the approximation quality.

11.4.3 Normal Approximation:

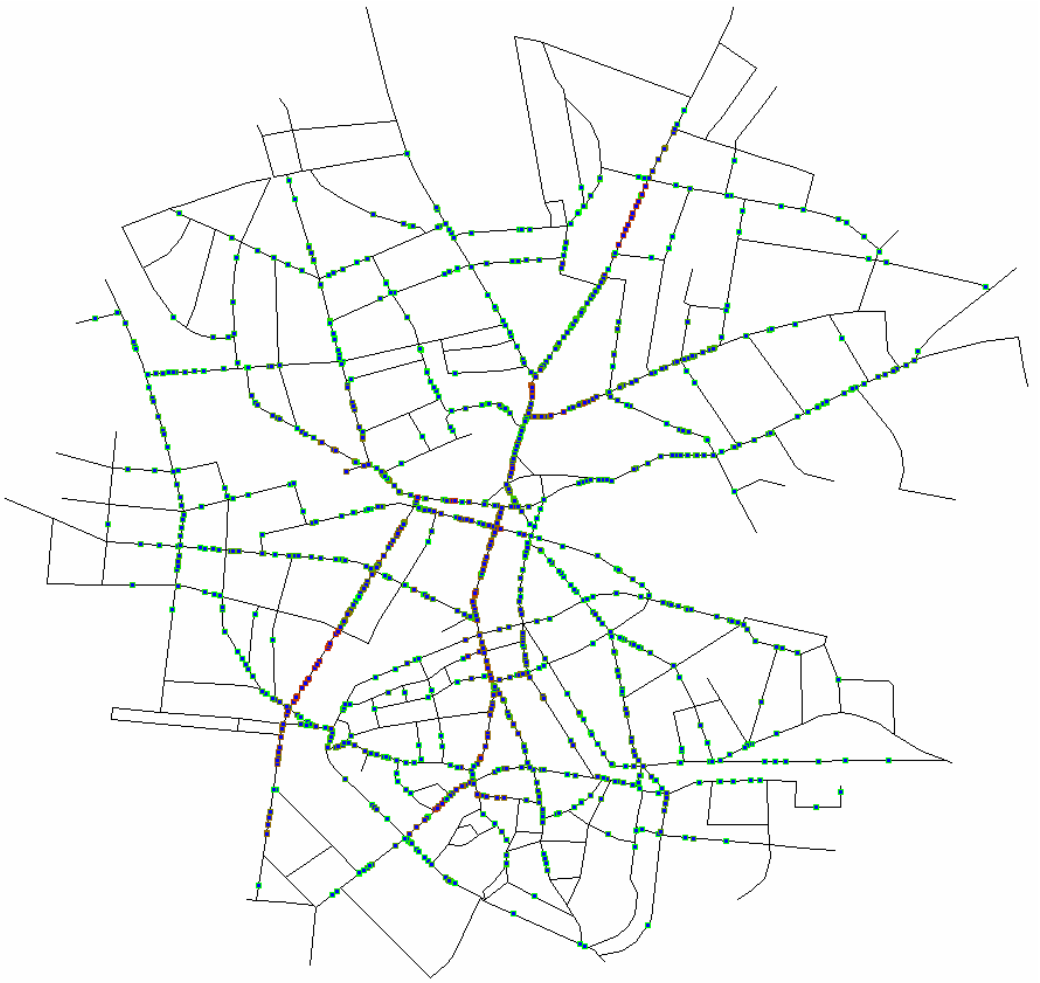
The implementation must make sure to apply continuity correction to acquire the best results. Pruning cannot be performed so that for each item set, the whole database has to be scanned. The experiments have shown that pruning only increases the total runtime marginally. Regarding the approximation quality, the normal approximation overall yields by far the the best results. Even for a small number of Poisson trials, the Normal approximation yields highly accurate results. On real data sets, it is very difficult to create a scenario where the normal approximation does not yield precision and recall values of one.

In summary, we propose to generally use the Normal approximation, except in very special settings, since the Normal approximation yields the best trade off between approximation quality, which is nearly always one, and efficiency, which is in $O(n)$ like the other approximation approaches.

In the future, we will examine how the model-based approach can be used to handle other mining problems (e.g., clustering) in uncertain databases. We will also study how other approximation techniques, such as sampling, can be used to mine PFIs.

Part V

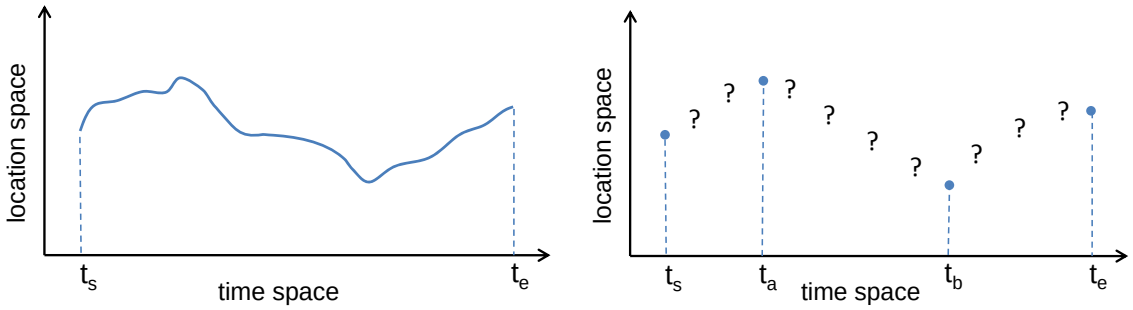
Querying and Mining Uncertain Spatio-Temporal Data



Efficient management of large collections of (location, time) data pertaining to mobile entities whose whereabout changes over time is a paramount in a plethora of application domains: from structural and environmental monitoring and weather forecasting, through disaster/rescue management and remediation, to Geographic Information Systems (GIS) and Tourist Information-Providing (TIP) systems. The technological enabling factors for such applications were advances in sensing and communication/networking, along with the miniaturizations of the computing devices and development of embedded systems. With the wide availability of satellite, RFID, GPS, sensor, wireless, and video technologies, spatio-temporal data, that is data containing both location and time information, has been collected in massive scale and is becoming increasingly large, rich, complex, and ubiquitous. In addition to the fact that in-between the discrete instances, some form of interpolation is needed, the measurements are imprecise, due to the physical limitation of the devices. From a complementary perspective - to reduce the communication and bandwidth utilization, along with the storage requirements, often the data is subjected to a reduction, thereby eliminating some of the known/recorded values. These issues are introducing the notion of uncertainty in the context of spatio-temporal data management - an aspect raising an imminent need for scalable and flexible data management.

In this field, a typical data source context consists of heterogeneous sensor deployments, including mobile stations, weather stations, satellite imagery, weather radar, mobile weather radar, stream observations, citizen supplied (*crowd sourced*) observations, ground and aerial LIDAR, water quality sampling, soil sampling and many more. In addition to many different types of sensors, the same type of sensor is often used redundantly, to measure the same variable from different positions and angles. It is clear that different sensors may yield inconsistent and contradictory information. Traditional database approaches *repair* [12, 33, 198] such inconsistencies by removing objects or by changing attribute values. Such approaches that discard uncertainty information cannot be applied here: Due to the uncertainty, the fraction of the *world* that is modelled by the database no longer has one single correct version, but instead, there exist many possible worlds, each associated with a probability of being true. Any single world, even the most likely world, resulting from a database repair may have very small probability of being true, due to the huge amount of possible worlds.

This trend towards big sensor environments, possibly using different sensing technologies to measure an environmental parameter, is emphasized by so-called *Next-Generation Sensor Networks*, described in the book "The Fourth Paradigm: Data-Intensive Scientific Discovery‘" ([120]). Such sensor environments have become possible due to the rapid decrease in the cost to produce sensor devices that become smaller and "smarter" and have lead us into a new age of ubiquitous sensing. In accordance to the technical concept of smart environment, there is an increasing trend to exploit the information derived from sensors and computational elements that are embedded in the objects of our lives and that are connected through a continuous network. The rapidly increasing storage capacity and communication bandwidth of such sensors and computational elements, facilitate to capture, exchange and store information about phenomenons or properties of our environment in unprecedented rates and variability. This evolving technology allows us to analyze such



(a) Ideal Case (b) Practical Case

Figure 11.8: Spatio-Temporal Data

data yielding new opportunities, and become useful, for many scientific and industrial applications such as location-based services, traffic monitoring and analysis, transportation management, environmental monitoring and analysis, among many others.

Managing, querying and mining spatio-temporal data has received a large amount of research interest in the past years. In most of these works however, the assumption is made that the trajectory, i.e., the function of a spatio-temporal object that maps each point in time to a position in space, is known entirely without any uncertainty, such as the trajectory depicted in Figure 11.8(a). A survey on techniques for managing, querying and mining trajectory data without uncertainty is given in [216]. However, it is not viable to maintain positions of moving objects continuously for each point in time for a series of reasons:

- Storing arbitrary continuous functions at high accuracy would require a very high space complexity, thus having detrimental impact on query performance.
- In most applications, measurements and observations are taken at discrete times only, e.g. due to limited bandwidth or energy constraints (e.g. in applications using GPS technology), thus the information of the full trajectory is not available.
- Some systems only allow to track the position of an object at predefined spatial positions (e.g. systems using RFID technology).

For practical applications it is thus mandatory to consider uncertainty in spatio-temporal data, such as depicted in Figure 11.8(b). As an exemplary application, consider the problem of monitoring iceberg activity in the North Atlantic. Ships transiting between Europe and east coast ports of North America traverse a great circular route that brings them into the vicinity of icebergs carried south by the cold Labrador Current near the Grand Banks. It was here that the R.M.S. Titanic sank in 1912, after it struck an iceberg. This disaster resulted in the loss of 1517 lives and led directly to the founding of the The International Ice Patrol (IIP) in 1914. The mission of the IIP is to monitor iceberg danger near the Grand Banks of Newfoundland and provide the locations of all known ice to the maritime community. The IIP does this by sighting icebergs, using visual observations from ships and

aircrafts, as well as data from buoys and radars. A database stores the recorded positions and extents of observed icebergs and data models are used to predict their movement, based on the uncertainty of the recorded observations. Estimating the position of an iceberg at a time t underlies two sources of error:

- the observation measurement error and
- the obsoleteness of the most recent observation.

Ignoring the uncertainty in such data, e.g., by treating the expected position of an iceberg as truth, may lead to disasters when icebergs deviate from the expected position, running into ships that have a false feeling of safety.

The aim of this part of this thesis is to uncover the potential knowledge contained in spatio-temporal data. This challenge is approached by dividing it into the following chapters:

- Chapter 12 gives necessary background knowledge and motivates the explicit consideration of uncertainty in spatio-temporal data. This chapter will review existing work on modeling uncertainty in spatio-temporal data and presents the *data model* upon which the query and mining algorithms presented in this thesis is based on. The presented model aims at maximizing the amount of information captured in the model, while at the same time providing the tools for efficient querying and mining algorithms. This section presents existing results borrowed from the statistics field of *stochastic processes* and puts these into the context of efficient spatio-temporal data management.

- Chapter 13 presents efficient algorithms for the problem of answering spatio-temporal window queries. By applying the paradigm of equivalent worlds (c.f. Chapter 3), this approach is the first one to answer such queries efficiently, while adhering to possible world semantics. These research results have been published at the IEEE International Conference on Data Engineering (ICDE) 2012 as a full paper ([66]).

- Chapter 14 studies the problem of answering spatio-temporal nearest neighbor queries. For this purpose, state-of-the-art nearest neighbor semantics defined on trajectories are generalized to the case of uncertain spatio-temporal data. This chapter theoretically investigates the complexity of the answering queries using these semantics, and comes to the conclusion that exact query processing all nearest neighbor semantics lead to NP-hard problems. The hardness of this class of problem motivates the use of approximate solutions as presented later in Chapter 16.

- To support the task of querying uncertain spatio-temporal data, effective spatio-temporal access structures are mandatory to avoid scanning the full database for each query. Chapter 15 therefore presents a first approach to effectively index spatio-temporal data based on the data model introduced in Chapter 12. Furthermore, Chapter 15 shows how to utilize the presented index structure to answer probabilistic spatio-temporal window queries and spatio-temporal nearest-neighbor queries as

defined in Chapter 13, speeding up such queries by orders of magnitude. This index structure has been published at the 21st ACM International Conference on Information and Knowledge Management (CIKM) 2012 as a full paper ([66]).

- To efficiently answer any type of spatio-temporal query in general, a sampling approach based on Bayesian learning is presented in Chapter 16. This approach allows to efficiently compute approximate answer for computationally hard query predicates. This chapter shows how traditional Monte-Carlo sampling fails in the context of spatio-temporal data, and proposes new techniques to adapt the data model to support efficient sampling. The research results presented in Chapter 14 and Chapter 16 have been accepted for publication at the 40th International Conference on Very Large Data Bases (VLDB) 2014 as a full paper ([141]).

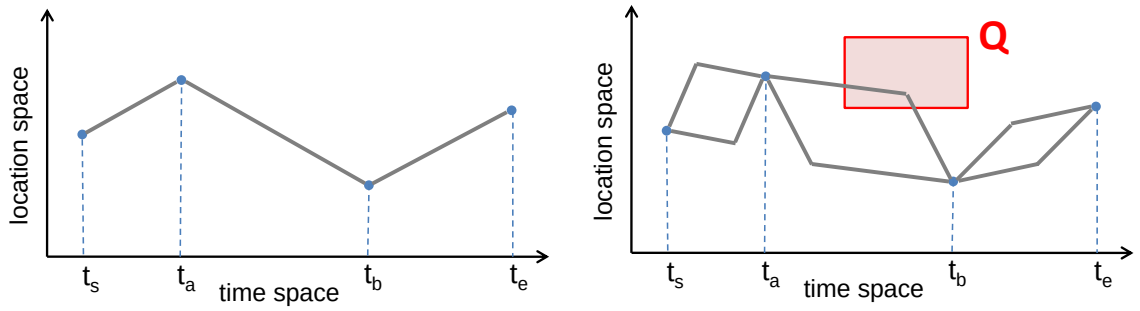
- Chapter 17 presents an experimental evaluation of efficiency and effectiveness of the spatio-temporal query types presented in Chapter 13 and Chapter 14 as well as an evaluation of the spatio-temporal access method presented in Chapter 15. Approximate solutions using the sampling based approach of Chapter 16 are presented as well. Furthermore, the effect of combining both the presented index structure (Chapter 15) and the proposed sampling approach (Chapter 16) is evaluated to learn the impact of these techniques on both efficiency and effectiveness of the kNN query predicate presented in Chapter 14. Some of the experimental results found in this chapter can be found in the aforementioned publications.

- Chapter 18 tackles the challenge of mining uncertain spatio-temporal data. Thus, in this chapter the challenge is to extract useful and previously unknown knowledge that is not explicitly stored in the database. As an example of a data mining application, Chapter 18 shows how the model presented in Section 13 can be utilized to predict traffic jams on a road network. The results presented in this chapter have been published at the SIAM International Conference on Data Mining (SDM) 2008 as a full paper ([113]).

Chapter 12

Modeling Uncertain Spatio-Temporal Data

In this thesis, the expression *spatio-temporal data* refers to a collection of data triples having the form (*Object-ID*,*Location*,*Time*). In practice, such tuples correspond to observations (e.g. GPS positions, RFID signals, or visual observations) of *moving objects* in the past. Thus, each tuple corresponds to a single point of the *trajectory* of a moving object. Many application using modern tracking technology create a flood of such *historical spatio-temporal data* ([136]). In practice, the position of an object is observed at discrete times only, leading to an inherent uncertainty between these discrete times. This leads to the notion of *uncertain spatio-temporal data*. Moving objects, for which the position at any time cannot be determined deterministically is denoted as a *uncertain moving object*. The indeterministic trajectory of an uncertain moving object is an *uncertain trajectory*. This following section, Section 12.1, reviews current state-of-the-art models to represent uncertain spatio-temporal data. Shortcomings of these models are addressed in Section 12.2, by applying models from statistics, namely stochastic processes, to treat uncertain moving objects in a probabilistic way.



(a) Linear Interpolation

(b) Spatio-Temporal Space Approximation

Figure 12.1: Interpolation between observations

12.1 State-of-the-Art

A large body of research has addressed the problem of modeling uncertainty in relational and spatial data, as surveyed in Section 2.2. In contrast, the problem of modeling and managing uncertain data with a temporal component has only received limited attention by the community. Most of the existing works treat spatio-temporal uncertainty as a simple extension of spatial uncertainty. For instance, the methods of [51] and [190] assume that for each timestamp in the history and every object, there is a location sample, which carries uncertainty. The samples are then modeled as uncertain regions, using probability density functions (PDFs) and these regions are connected to form the trajectories. Given a spatio-temporal range query [51], we can then estimate the probability that a trajectory intersects the window by the overlap of the corresponding PDFs with the window. These types of models, however, may not be acceptable in the case where we only have a sample of locations in the moving history of an object. In this case, for timestamps where locations are not sampled, we have to infer the whereabouts of the object with the help of stochastic models. These models are particularly useful when predicting the future locations of objects, with the help of current (or recent) location and movement observations. In addition, these models allow for the economic representation and storage of the data, since only a subset of the object locations need to be sampled and used for the inference of their remaining locations.

12.1.1 Interpolation Models

In recent years, the research community often employed linear interpolation to model the movement of spatio-temporal objects [151, 160, 183], as depicted in Figure 12.1(a). Based on this model, [179] and [181] addressed the problem of query processing (window queries, joins, and nearest neighbour queries) under the assumption of linear interpolation. With the algorithms introduced in these works, the validity interval of a query result can be computed. However, clearly, the simple model of linearly interpolating between observations shows several drawbacks. On the one hand, whenever the sampling rate becomes low, the true trajectory of an object might deviate greatly from the estimated trajectory. On the other hand it is even possible that a linearly interpolated trajectory becomes impossible: Imagine a car travelling from Munich to Berlin with sample points in exactly those two cities. The driver will never travel exactly along the straight line between the two cities because there are no streets following this trajectory. Although linear interpolation is one of the most common means of modeling the motion between observations in certain databases, other approaches have been evaluated as well. Later, [178] introduced a framework that allows the future motion of objects to be described in a more complex manner than linear interpolation. Based on these complex motion functions, the authors aimed at predicting the future position of objects with minimized error. All approaches introduced so far do not take uncertainty into account. However, in scenarios where data are inherently uncertain, such as spatial and spatio-temporal databases, answering traditional queries using expected values and positions is inadequate, since the results could be

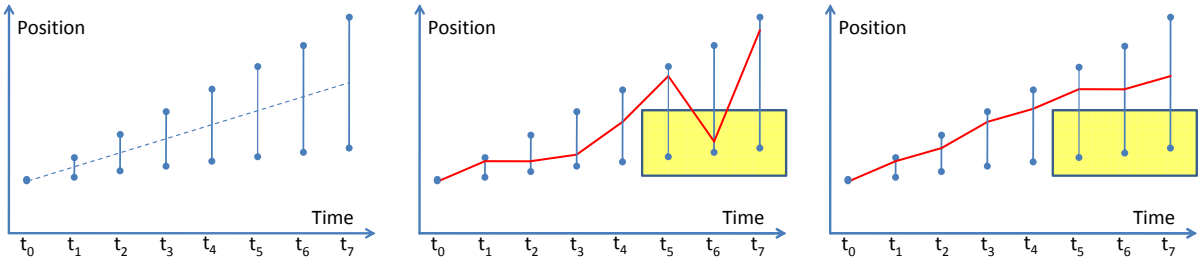
incorrect [31]. To mitigate this problem, several algorithms taking uncertainty into account have been developed. Conservative Space-Time Approximation Models: The prevalent approach is to bound all possible (time, location) pairs of an object by a simple geometric structure in time and space. The result is a spatio-temporal approximation that is based on previous knowledge such as the maximum speed and the maximum acceleration of objects. An example for the one-dimensional case is shown in Figure 12.1(b), using the maximum speed of objects to obtain a conservative two-dimensional (space, time)-approximation. Generally, such approaches utilize various geometrical shapes, such as sheared cylinders [190, 191, 192], diamonds [139] and so called beads [116, 188] for the case of two spatial dimensions (the third dimension is time). Queries on these models include range queries [150, 192, 191] and kNN queries [190, 51]. The main problem of all these approaches is that no probability information is given for any object approximation. Thus, it is not possible to assess the probability of an object to satisfy some query predicate: For example, for the query window Q depicted in Figure 12.1(b), the only conclusion that can be made is that the approximated object may possibly intersect Q. However no information about the likelihood of this event can be assessed. In particular, this probability can be zero due to the conservative nature of these approximation models. Simple assumptions to estimate this probability, such as a uniform distribution over the conservative approximation, are often impractical: In practice, a vehicle having a fairly constant velocity between two (location, time) pairs is more likely than the same vehicle going at maximum speed, passing its destination, then performing a U-turn to race back the opposite direction in order to barely reach the second (location, time) pair in time.

12.1.2 Models ignoring time dependencies

In order to assess the actual probability of events in a spatio-temporal database, Mokhtar and Su [139] describe a model where the uncertainty region of each object is described by a time dependent stochastic process. Objects are given by MBRs which change their location and extent over time following the stochastic process. The paper shows how to answer certain types of window queries based on this model. However, describing the parameters of the uncertainty regions (instead of the object trajectories) by a stochastic process yields wrong results, i.e., results that cannot possibly be the correct result. The reason is that location dependency between consecutive timestamps is ignored by this model. Hence the position of an object at time t is assumed to be independent of its previous position at time $t-1$.

Approaches like [14] and [201] consider uncertain time series (where the concrete value at each point of time is not known for certain) and data streams, respectively. Similar to other work, they disregard correlations between points in time.

To illustrate the problem of these approaches disregarding temporal dependencies, consider Figure 12.2(a), where an uncertain object trajectory is modelled. Here, it is assumed that an object o moves forward in a one-dimensional space with an uncertain velocity. The velocity of o may change over time, but will not drop below some minimum speed greater than zero, and will not exceed some maximum speed. Therefore, given the position of o



(a) Abstract Model (b) Assuming Temporal Independence (c) No Temporal Independence

Figure 12.2: Modeling Spatio-Temporal Data

at time t_0, the future position of the object can be modelled using the expected speed of o, depicted by the dashed line in Figure 12.2(a), as well as lower and upper bounds, or alternatively a variance, depicted by the intervals at each point of time. Under the assumption of temporal independence, at each point of time, the positions of o are modelled as independent random variables. Therefore, a trajectory as depicted in Figure 12.2(b) has a probability greater than zero. However, since o makes a large leap backward between times t_5 and t_6, this trajectory is not possible given the knowledge about the movement of o, which this model should incorporate. A possible trajectory is shown in Figure 12.2(c). Here, the object moves within its speed limits at each point of time.

The flaw of modeling trajectories which are not actually possible becomes a problem when processing spatio-temporal queries based on this model. For example, consider a spatio-temporal window query, which returns for an object o the probability that o intersects the query window q, depicted in Figure 12.2(b) and Figure 12.2(c). For any model that ignores the dependency between locations at subsequent points of time, the probability that o is always outside of the window is the product of many probabilities, thus becoming very small. Therefore, for a large number of points of time inside the query region, the probability that o intersects the query window converges to one. However, if the dependency between locations at subsequent points of time is considered, then the probability that o is outside of the window at time t_6 depends on the probability at time t_5 in this example: If o is not in the window q at t_5, then it cannot be in q at t_6 either, since the object cannot move backwards. Thus, the probability that o intersects the query window q at any time, is equal to the probability that o intersects the query window at time t_5. Therefore, an important aim in this chapter is to properly model such dependencies, instead of simply treating time as an additional dimension in space. Models considering time dependencies: An initial approach to address the problem of temporal dependencies has been made in [154]. The authors assume a discrete state space and the Markov property to allow transitions between successive points of time. Their query language Lahar allows us to formulate queries by stating regular expressions on an alphabet of states, and returns the probability of observing a sequence of states satisfying this regular expression. However, this syntax cannot handle time context as required by many common queries.

For example, no regular expression can express the language that contains at least one character x at a given position interval. Such a query corresponds to a window query as used above. Chapter 13 presents a framework for efficiently modeling and querying uncertain spatio-temporal data that, in contrast to existing research, take time dependencies into account.

12.2 Modeling Uncertain Spatio-Temporal Data

The key idea of the proposed approach is, similar to [154]: to model possible object trajectories by stochastic processes, more precisely a Markov chain. Employing the Markov chain model for representing spatio-temporal data has three major advantages over previous work:

1 It allows answering queries such that results are associated with corresponding probabilities.

2 Dependencies between object locations at consecutive points in time are taken into account.

3 It is possible to reduce all queries on this model to simple matrix operations, because transitions between spatial entities over time can be performed by matrix multiplications, for which there exist efficient solutions.

Following state-of-the-art models, discrete space and time domains S and $\mathcal{T}$ are assumed, where space is defined by coordinates and time denotes points in time. In practice, discretization techniques (e.g. a unidistant grid) can be used to satisfy this assumption.

Formally, let $S = \{s_1, ...s_{|S|}\} \subseteq \mathbb{R}^d$ be a finite set of possible locations in space which we call *states* and let $\mathcal{T} = \mathbb{N}_0^+$ be the time-space. Consequently, a (certain) object o that moves in space is represented by a *trajectory* given by a function $o : \mathcal{T} \to S$ that defines the location $o(t) \in S$ of o at a certain point of time $t \in \mathcal{T}$.

Definition 53 (Spatio-Temporal Object)**.** *Given a d-dimensional spatial domain $\mathcal{S}$ and a time domain $\mathcal{T}$, a spatio-temporal object comprises a set Θ^o of pairs $\Theta_i^o = (\Theta_i^o.s, \Theta_i^o.t) \in \mathcal{S} \times \mathcal{T}$.*

Semantically, each pair $\Theta_i^o \in \Theta^o$ corresponds to the observation that o is located at location $\Theta_i^o.s$ at time $\Theta_i^o.t$. The observation $argMin_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t)$ is called the *first observation* of o, and the observation $argMax_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t)$ is called the *last observation* of o. The time interval $[argMin_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t).t, argMax_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t).t]$ defined by the times of the first and last observation, is called the *lifespan* of o.

A main challenge of such data, is to infer the position of o at a time $t : \neg\exists s : (s, t) \in \Theta^o$ for which the exact position has not been observed. The quality of this inference depends on our ability to effectively use the wealth of information stored in a spatio-temporal database, including

- all observations temporally preceding (i.e., in the past of) o,
- all observations temporally succeeding (i.e., in the future of) o,
- information about possible trajectories between observations, e.g. given by a road network and
- empiric information about movement patterns, such as turn probabilities at a road intersection, empirically learned from other objects.

Following the tradition of uncertain spatial non-temporal databases, we suppose that the locations of an uncertain spatio-temporal object $o \in \mathcal{DB}$ at time t are realizations of a random variable $o(t)$. This model implies that a spatio-temporal objects is described by a series of random variable $(o(t))_{t \in T}$. This consideration directly equals the definition of a *stochastic process* [101]:

Definition 54 (Stochastic Process)**.** *A stochastic process X is a family of random variables $(X(t) \in S)_{t \in \mathcal{T} \subseteq \mathbb{R}}$. If $T \subseteq \mathbb{N}$, we say that X is a discrete stochastic process.*

Note that in general, these random variables $X(t_1), X(t_2), t_1, t_2 \in T$ are stochastically dependent.

Example 27. *As a simple example for a first stochastic process, let $D_1, ..., D_n$ be a series of random variables such that D_i is the result of a single throw of a six-sided dice. Let $X^1(t), 1 \leq t \leq n$ be a stochastic process, such that $X(t) = \sum_{i=1}^{t} D_i$ is the total sum of pips shown after t dice throws. A possible realization of this random variable is $(3, 8, 9, 11, 15, 21, 26, ...)$. Clearly, each random variable $X(t)$ is stochastically dependent of the random variable $X(t-1)$, as the state space of $X(t)$ is bounded by $[X(t-1)+1, X(t-1)+6]$. Due to the same observation, $X(t)$ is stochastically dependent of $X(t+1)$, yielding transitive (by induction) dependencies between all random variables $X(t_1)$, $X(t_2)$ where $t_1, t_2 \in [1, ..., n]$.*

Definition 55 (Uncertain Object Trajectory)**.** *Given the spatial domain $\mathcal{S}$ and the time domain $\mathcal{T}$, an* uncertain object trajectory *$o(t) \in S$ of an object $o \in \mathcal{DB}$ is a stochastic process $(o(t) \in S)_{t \in \mathcal{T}}$.*

An example of an uncertain object trajectory of an object $o \in \mathcal{DB}$ is illustrated in Figure 12.3. The raster models all possible locations (i.e., states) in S, shown for the time sequence $\langle t_0, \ldots, t_3 \rangle$. Here we assume that object o has been obsered at time t_0; all locations of o that follow are uncertain. Consequently, the uncertain trajectory of o comprises all possible trajectories starting at $o(t_0)$.

In accordance to the previous section, the constituent parts of an uncertain spatio-temporal object are specifications of probability distributions over the space and time domain. Naive models typically restrict the regions that the object can be at each timestamp. All uncertain trajectories are combinations of locations in these regions, giving all these trajectories equal probabilities ([190, 192, 191, 14, 201, 139]). However, as already mentioned, these models disregard any time dependencies between locations, treating each random variable $o(t)$ as independent.

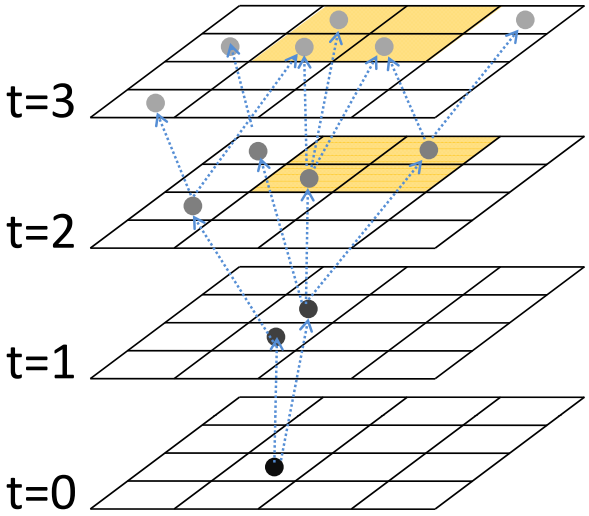


Figure 12.3: Querying Uncertain Spatio-Temporal Data

Example 28. *To illustrate the problem of discarding stochastic dependencies reconsider the throw-a-dice example in Example 27, where the random variable $X(t)$ corresponds to the total number of pips after t dice throws. Discarding stochastic dependencies between these random variables, each $X(t)$ is drawn independently. Thus, a possible instantiation of X is $(3, 7, 14, 9, 14, 29)$. Clearly, this series is impossible, as e.g., the number of total pips after three dice throws cannot equal 14, given that the first two dices had a sum of 7 pips. Nor are negative numbers of pips possible.*

In a spatio-temporal database, where the random variable is the position of an object, the effects of ignoring stochastic dependence between positions at different times leads to the effects we have seen in the example of Figure 12.2. In this example, ignoring dependencies leading to possible trajectories having an impossibly high velocity in either direction.

According to Definition 55, the uncertain motion of an object is defined as a stochastic process. In this work, the motion of moving objects through space and time is model by a first-order Markov-Chain model. Formally:

Definition 56. *A stochastic process $o(t), t \in \mathcal{T}$ is called a Markov-Chain if and only if*

$$\forall t \in \mathbb{N}_0 \forall s_j, s_i, s_{t-1}, ...s_0 \in S :$$

$$P(o(t+1) = s_j | o(t) = s_i, o(t-1) = s_{t-1}, ..., o(0) = s_0) =$$

$$P(o_{t+1} = s_j | o_t = s_i)$$

The above equality is called Markov assumption or Markov property. The conditional probability

$$P_{i,j}(t) := P(o(t+1) = s_j | o(t) = s_i)$$

is the (single-step) *transition probability* of state s_i to state s_j at time t.

In this work we assume that the transition probabilities $P_{i,j}(t)$ are given, e.g. derived from expert knowledge or derived from historical data. For example, the current of water in the Atlantic ocean can be used to infer the transitions of icebergs. For traffic data, the transition probabilities at road intersections can be estimated using historical data [47].

Definition 57. *A Markov Chain is homogeneous if and only if the transition probabilities are independent of t, i.e. $P_{i,j}(t) = P_{i,j}$.*

The (single-step) transition matrix $M = P_{i,j} \in \mathbb{R}^{S^2}$ is a *stochastic* matrix, i.e. the following properties hold:

$$\forall i, j \in S : P_{i,j} \geq 0$$

$$\forall i \in S, \sum_{j \in S} P_{i,j} = 1$$

Let $P(o,t)$ be the distribution vector of an object o at time t, such that $(P(o,t) = s_1), \ldots, P(o,t) = s_{|\mathcal{S}|})$, $p_i \in P(o,t)$ corresponds to the probability that o is located at state s_i at time t. The distribution vector of o at time $t+1$ can be inferred from $P(o,t)$ as follows:

Corollary 10.

$$P(o, t+1) = P(o,t) \cdot M$$

The *m-step transition probability* $P^t_{i,j}$ is the probability that an uncertain object o that is located at state s_i at time t, will be located at state s_j at time $t+m$ and can be computed exploiting the Equations of Chapman-Kolmogorov ([145]) as follows:

$$(P^m_{i,j}) = M^m$$

Given the probability distribution $P(o,t)$ of an uncertain object o at time t, the probability distribution $P(o,t+m)$ of o at time $t+m$ can be computed by

Corollary 11.

$$P(o, t+m) = P(o,t) \cdot M^m$$

To estimate the location of an object o, Corollary 10 and Corollary 11 require the position of an object o at some time in the past. In a nutshell, the Markov-model allows to carry information about the position of an object o at some time t over to future points of time $t+k$. The quality of this information, and thus the predictive power of the model, diminishes as the time horizon k increases. This fact is attributed to cumulative prediction errors incurred over time. In particular, if k approaches infinity, the predicted location distribution approaches the *stationary distribution* of the Markov model, which is independent of the location of t. To ease this loss of information, a main challenge of this thesis is to use all observations of an object for accurate location prediction. This task will be achieved by employing a Bayesian learning approach to adapt transition probabilities of an object given all observations in the past, in the current, and in the future.

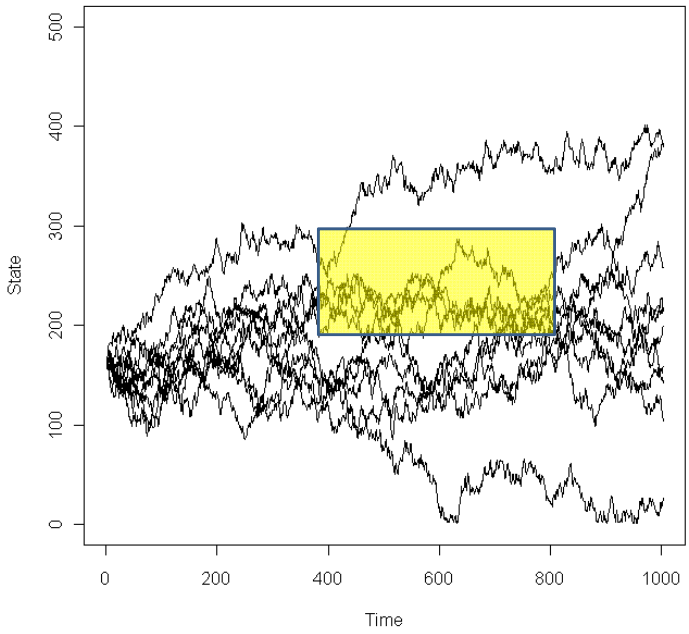


Figure 12.4: Some possible worlds of one uncertain object

Definition 58 (Trajectory Probability)**.** *Let $M(t)$ be a Markov chain and let $r = (s_1, ..., s_{|T|})$ be a trajectory of length $|T|$. The probability $P(o = r)$ that an object o takes trajectory r at a time t is given by the product between the probability that o is located at s_1 at time t, and the probabilities of all transitions of this trajectory.*

$$P(o = r, t) := P(o(t) = s_1) \cdot \prod_{i=1}^{|T|} \cdot M_{s_i s_{i+1}}(i)$$

To answer spatio-temporal queries on top of this model the main challenge is to correctly consider the possible world semantics in the model. The set of possible worlds of a spatio-temporal object o is spanned by the set of all possible trajectories of o.

Definition 59 (Possible Trajectories of an Object)**.** *Let $M(t)$ be a Markov chain and let $T = [t_{start}, t_{end}]$ be a time interval. The set*

$$\mathcal{W}(o, T) := \{(s_1, ..., s_{t_{start} - t_{end} + 1}) | P(o = (s_1, ..., s_{t_{start} - t_{end} + 1}), t) > 0\}$$

is called the set of possible trajectories of o during T. The set

$$\mathcal{W}(o) := \mathcal{W}(o, o.T = [argMin_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t).t, argMax_{\Theta_i^o \in \Theta^o}(\Theta_i^o.t).t]),$$

of possible trajectories of o during o's whole lifespan is called the set of possible trajectories of o.

Clearly, the number of possible trajectories of an object o increases exponential in the lifespan of o: Even in the case where each state s_i only has two possible subsequent states,

i.e., the case where each line M_i of the Markov chain has exactly two non-zero elements, the number of possible trajectories of o equals $2^{|o.T|-1}$.

Definition 59 defines the set of possible worlds of a single database object o. Similar to the case of spatial non-temporal data, the set of all possible worlds is defined by all combinations of possible instances of individual objects.

Definition 60 (Possible Worlds)**.** *Let $\mathcal{DB}$ be a spatio-temporal data storing observations of objects $o_1, ..., o_N$. The set of possible worlds $\mathcal{W}$ is defined as*

$$\mathcal{W}(\mathcal{DB}) = \{(r_1, ..., r_N) | r_i \in \mathcal{W}(o_i)\}.$$

The total number $|\mathcal{W}|$ of possible worlds equals the product $\prod_{i=1}^{N} |\mathcal{W}(o)|$. Since the number of worlds $|\mathcal{W}(o)|$ is in $O(2^{|T|})$, where T is the average lifespan of an object, the total number of possible worlds is in $O((2^T)^N) = O(2^{T \cdot N})$.

To evaluate a given query predicate ϕ on $\mathcal{DB}$, possible world semantics require to evaluate ϕ on each possible worlds, to compute the probability that $\mathcal{DB}$ satisfies ϕ.

$$P(\phi, \mathcal{DB}) = \sum_{w \in \mathcal{W}(\mathcal{DB})} \mathcal{I}(\phi, w).$$

This equation is identical to Equation 2.3 used for spatial non-temporal data in Chapter 2. The sole exception in this definition is that a world is now given by a set of object trajectories, rather than by a set of object instances. To avoid the enumeration of this double exponential set of possible worlds, the following chapters will once more apply the paradigm of equivalent worlds, to develop efficient query processing algorithms.

The next chapter, Chapter 13 presents a novel approach to efficiently answer spatio-temporal window queries. This type of query can exploit stochastic independence assumed between moving objects to find efficient algorithms. The problem of nearest neighbor search on uncertain spatio-temporal data is tackled in Chapter 14. In the case of nearest neighbor queries, objects do influence each others probability to be the nearest neighbor of a query trajectory. This chapter will define different types of nearest neighbor semantics, and show which semantics lead to exponentially hard query processing. For semantics having an efficient solution, efficient algorithms are presented. A spatio-temporal index structure is presented in Chapter 15, which supports both spatio-temporal window queries defined in Chapter 13 as well as spatio-temporal nearest neighbor queries defined in Chapter 14. To allow answering any general type of spatio-temporal query, a sampling approach based on Bayesian learning is presented in Chapter 16. This approach allows to efficiently compute approximate answer for hard query predicates, such as some of the nearest neighbor semantics presented in Chapter 14. It is shown how traditional Monte-Carlo sampling fails in the context of spatio-temporal data, and proposes new techniques to adapt the Markov-chain to support efficient sampling.

A thorough experimental evaluation of all proposed spatio-temporal query types, the impact of the proposed index structure, and the sampling approach will be given in Chapter 17.

Chapter 13

Spatio-Temporal Window Queries

The main contribution of this chapter is a framework for processing spatio-temporal queries on uncertain spatio-temporal data. This framework relies on efficient analysis of the space of possible worlds by multiplying Markov-Chain transition matrices. This framework exploits the power of existing tools for matrix multiplication, e.g. provided by Matlab, in order to accelerate the computation of spatio-temporal data distributions in accordance to possible worlds semantics. We gracefully integrate pruning approaches into the Markov Chain matrices, which results in drastically reducing the search space and the computational effort during query evaluation. The proposed framework is general enough to be applied for cases where an arbitrary number of observations exist for uncertain moving objects and for different spatio-temporal query variants as we demonstrate in Sections 13.3 and 13.4, respectively. As we show experimentally later, in Chapter 17, we achieve a speed up of multiple orders of magnitude compared to a straightforward solution, which relies on Monte-Carlo simulation over the space of possible trajectories that the objects may follow. Our approach is easily implementable because the tools provided by Matlab libraries are available for all common programming languages (e.g., C/C++, Java, etc.) and, as a result, they can be easily integrated into existing uncertain DBMSs.

13.1 Problem Definition

Our goal is to efficiently evaluate probabilistic spatio-temporal queries on uncertain spatio-temporal objects; i.e., queries about objects that are probably located in a given spatial region during a given range in time. Within the scope of this chapter, we focus on spatio-temporal queries specified by the following parameters: (i) a spatial region $S^{\Box} \subseteq S$, i.e. a set of (not necessarily connected) locations in space, and (ii) a set $T^{\Box} \subseteq \mathcal{T}$ of (not necessarily subsequent) points in time. In the remainder, we use $Q^{\Box} = S^{\Box} \times T^{\Box}$ to denote the query ranges in the space and time domain. The most intuitive definition of a probabilistic spatio-temporal query window is given below:

Definition 61. *[Probabilistic Spatio-Temporal (Exists) Query] Given a query region $S^{\square}$ in space and a query region $T^{\square}$ in time, a* probabilistic spatio-temporal exists query *(PST∃Q), retrieves for each object $o \in \mathcal{DB}$ the probability $P(o(t) = s) \in [0, 1]$ that o is located in $S^{\square}$ at some time $t \in T^{\square}$.*

This query type has been studied before (e.g. in [192, 191]), albeit over data models that disregard dependencies between locations at consecutive timestamps, as we have discussed in Section 12. For our motivating application described in the previous chapter, an exemplary query could be: find all icebergs that have non-zero probability to be inside the movement range of a particular ship during the ship's movement in the North Atlantic. Another query could be to predict the number of cars that will be in a congested road segment after 10-15 minutes.

In addition, we study the following two interesting probabilistic query variants. Note that the second variant has not been considered in the past:

Definition 62. *[Probabilistic Spatio-Temporal For-All Query] A* probabilistic spatio-temporal for-all query *(PST∀Q) retrieves for each object $o \in \mathcal{DB}$ the probability $P(o(t) = s) \in [0, 1]$ that o remains in $S^{\square}$ for* all *times $t \in T^{\square}$.*

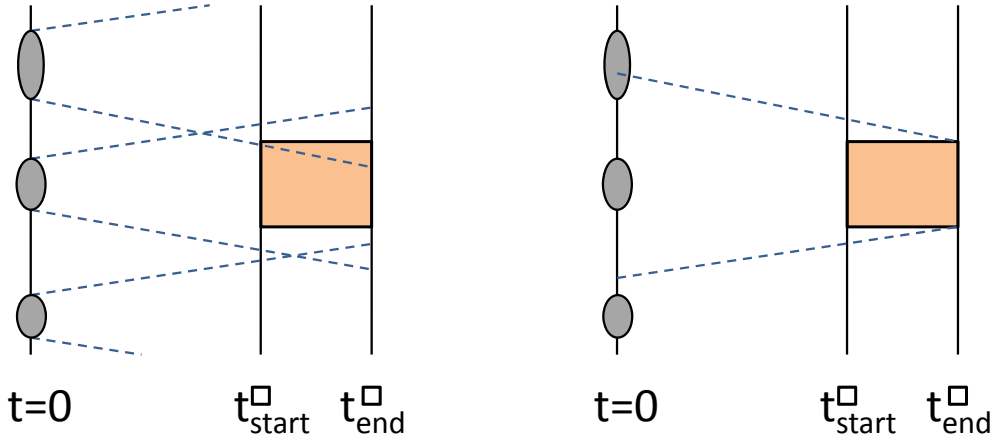
Definition 63. *[Probabilistic Spatio-Temporal k-Times Query] A* probabilistic spatio-temporal k-times query *(PSTkQ) retrieves for each object $o \in \mathcal{DB}$ and each parameter $1 \leq k \leq |T^{\square}|$ the probability that o is located in $S^{\square}$ at* exactly *k times $t \in T^{\square}$.*

PST∀Q and PSTkQ are important complements to the PS∃TQ. For example, these queries can progressively determine candidates that remain in a certain region for a while. For example, for a given region somewhere in the north Atlantic we want to retrieve all icebergs that have non-zero probability remaining in this region for a specified period of time, e.g. to be able to make some measurements over a certain time period. Further examples where such queries are useful are for location-based-service (LBS) applications, e.g. a service provider could be interested in customers that remain at a certain region for a while, such that they can receive advertisements relevant to the location.

Note that, although the spatial (temporal) parameters of the queries define contiguous regions (intervals) in the space (time) domain, our query processing approaches are also applicable for any arbitrary subset of the space (time) domain.

13.2 Probabilistic Spatio-Temporal Query Processing using the Markov-Chain Model

In this section, we show how the queries that we presented in Section 13.1 can be evaluated efficiently. For the ease of presentation, we first assume that for each object, there is a single observation at time $t = 0$ and that we want to predict the result of a probabilistic spatio-temporal exists query (Definition 61) in the future. We propose two approaches towards efficient query processing. The *object-based approach* (Section 13.2.1) directly computes



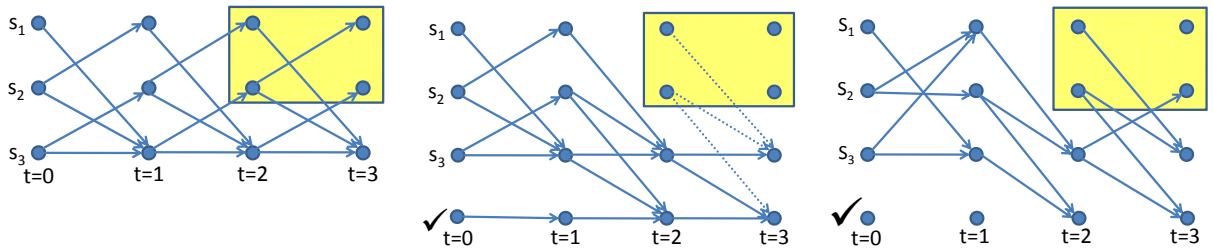
(a) Object based aproach (b) Query based approach

Figure 13.1: Schematic illustration of OB and QB

$P^{\exists}(o, S^{\square}, T^{\square})$ in an efficient way, while the *query-based approach* (13.2.2) follows a reverse methodology: computation starts at the query window and the transposed Markov-Chain matrix is used to compute, for each state $s \in \mathcal{S}$ the probability that an object starting at s satisfies the query predicate. We generalize our results for the case of multiple observations and other queries in Sections 13.3 and 13.4, respectively.

Figure 13.1 shows a high-level example of query evaluation using the object-based and the query-based approaches. Consider a spatio-temporal query with query time interval $[t^{\square}_{start}, t^{\square}_{end}]$ illustrated by the shaded rectangle on the right of each subfigure. We would like to predict the result of the query, based on observations about the locations of the objects at time $(t = 0)$ and a given model that captures their state transition probabilities (states correspond to spatial locations illustrated by the y-axis in the example). Exemplary objects are shown in oval shapes.

The object-based approach, illustrated in Figure 13.1(a), examines for each object the possible trajectories (i.e., possible worlds) that the object will follow in the future and finds the probabilities of trajectories that intersect the query window. For instance, the top-most object has a non-zero probability to be a result of the query, the middle object has a higher probability and the object at the bottom has a zero probability. To compute the probabilities of all possible worlds that intersect the window, for each object, we use matrix multiplications. We show how pruning techniques can be incorporated into the matrix multiplications for efficiency. Still, this approach iteratively examines all objects in the database exhaustively. The query-based approach (Figure 13.1(b)), on the other hand, *reverses* the computation: given that an object intersects the query window, we compute the probability that this object corresponds to any of the objects in the database; this way, examining objects that are irrelevant to the query is avoided, while at the same time the query results are computed in batch. We now describe these two approaches in detail.



(a) All possible worlds (b) Object-based approach (c) Query-based approach

Figure 13.2: Procedure of OB and QB

13.2.1 Object-Based Query Processing

Given an object o, in order to find the probability that o is part of a query result, we could use the following straightforward approach: compute, for each point of time $t \in T^{\square}$ the probability that o is located in $S^{\square}$ and aggregate these probabilities. Clearly, this approach yields incorrect results (for example, if $|T^{\square}| = 3$ and for each $t \in T^{\square}$,$P(o,t) = 0.5$). The problem of this approach is that a specific possible world (i.e., trajectory) may be considered more than once, if it overlaps with the query at multiple timestamps. Therefore, to correctly process queries, possible worlds which satisfy the query predicate should only be considered once in the computation of the result.

As an example of proper query evaluation, consider the Markov-Chain:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0.6 & 0 & 0.4 \\ 0 & 0.8 & 0.2 \end{pmatrix},$$

for which all possible worlds are depicted in Figure 13.2(a) for the first four points of time.[1] Additionally, assume a window query defined by $S^{\square} = \{s_1, s_2\}$ and $T^{\square} = \{2, 3\}$ and assume an object o which has been observed at s_2 at time $t = 0$, i.e. $P(o, 0) = (0, 1, 0)$. To compute the probability $P^{\exists}(o, S^{\square}, T^{\square})$ that o intersects the query window, we first compute the probability distribution $P(o, 2)$ at time $t = 2$, using Corollary 11. The resulting probability vector $P(o, 2) = (0, 0.32, 0.68)$ gives us a lower bound of 32% for $P^{o,\exists}(S^{\square}, T^{\square})$, since any world in which o is located at state s_2 at time $t = 2$ satisfies the query window. Thus, these worlds can already be considered as true hits and must be ignored at future points of time. Thus, we obtain a new probability distribution $P(o, 2)' = (0, 0, 0.68)$ which will be used for the next state transition using Corollary 10. The resulting vector $P(o, 3) = (0, 0.544, 0.136)$ means that out of the remaining 68% of worlds which have not already been reported as true hits, another 54.4% can now be returned as true hits, since in these worlds o is located at s_2 at time $t = 3$. Since $t = 3$ is the last point of time belonging to the query window, the fraction of 0.136 worlds can be reported as true drops,

[1] Probabilities are omitted for readability, but can be found in the Markov-Chain.

since in these worlds, o has not intersected the query window at any $t \in T^{\square}$. Thus, the result of this query is $0.32 + 0.544 = 0.864$.

The above example gives an intuition on how to answer queries correctly by identifying worlds that satisfy the query and excluding them from further processing. We incorporate this technique directly in the Markov-Chain, to facilitate efficient on-the-fly pruning when matrix multiplication is performed between a state vector and the Markov-Chain. To achieve this goal, we introduce a new state which is denoted by "✓", denoting true hits, i.e. for any world in which an object o reaches ✓, we know that o satisfies the query predicate. This ✓ state satisfies the *absorbing* property, i.e. any world reaching this state cannot leave it. Instead of directly using the Markov-Chain M, we build the following two new matrices

$$M^{-} = \begin{pmatrix} M & zero(|S|) \\ zero(|S|)^T & 1 \end{pmatrix}$$

and

$$M^{+} = \begin{pmatrix} M' & sum(S^{\square}) \\ zero(|S|) & 1 \end{pmatrix},$$

where $zero(|S|)$ is a vector of size S containing all zeroes, $zero(|S|)^T$ is its transposed, M' is derived from M by replacing all columns that correspond to states in $S^{\square}$ by zero vectors, and $sum(S^{\square})$ is a column vector containing for each line in M the sum of values removed this way.

The initial object distribution vector of an object o is now extended by an additional value of zero, corresponding to the fact that initially (at time $t = 0$) we cannot identify any worlds which satisfy the query predicate.[2] At each state transition, where the target state does not belong to $T^{\square}$, we can now use M^{-} instead of M, which has the same effect as M, while preserving the probability of state ✓. If the target state belongs to $S^{\square}$, then M^{+} is used instead. This way, worlds leading into states in $S^{\square}$ are now redirected to state ✓ instead.

Example 29. *For the matrix* $\begin{pmatrix} 0 & 0 & 1 \\ 0.6 & 0 & 0.4 \\ 0 & 0.8 & 0.2 \end{pmatrix}$ *of our running example, the corresponding new matrices are* $M^{-} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0.6 & 0 & 0.4 & 0 \\ 0 & 0.8 & 0.2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ *and* $M^{+} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 0 & 0 & 0.2 & 0.8 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

The corresponding visualization is depicted in Figure 13.2(b). Here M^{-} is used for the first transition from $t = 0$ to $t = 1$, while for the transitions from $t = 1$ to $t = 2$ and from $t = 2$ to $t = 3$, M^{+} is utilized as transition matrix. Thus, for an object that has been observed at state s_2 at time $t = 0$, we obtain the initial probability distribution vector $P(o, 0) = (0, 1, 0, 0)$, where the fourth value denotes the initial probability of being

[2] In the special case where $t = 0$ belongs to $T^{\square}$, we adjust the initial vector by moving all probabilities of states in $S^{\square}$ to state ✓.

a true hit, which is zero since $t = 0$ does not belong to $T^{\square}$. The transition to $t = 1$ yields $P(o,1) = P(o,0) \cdot M^{-} = (0.6, 0, 0.4, 0)$. Clearly, the fourth value corresponding to state ✓ is zero, since no state $t \in T^{\square}$ has been visited so far. Transition to $t = 2$ yields $P(o,2) = P(o,1) \cdot M^{+} = (0, 0, 0.64, 0.36)$ and the final transition yields $P(o,3) = P(o,2) \cdot M^{+} = (0, 0, 0.136, 0.864)$. Therefore, the resulting probability that o intersects the query window is 0.864.

13.2.2 Query-Based Query Processing

The object-based approach applies for each object o the methodology described in Section 13.2.1 to compute the probability that o is part of the query result. The query-based approach assumes that an object intersects the query. Based on this assumption, this approach starts at the last point of time in the query window, and goes backward in time using the transposed Markov-Chain. This way, we can compute, at any time t, the probability vector $P(t)$ containing for each state s, the probability that an object starting at state s at time t will satisfy the query predicate.

Therefore, we start at time $t^{\square}_{end} := max(T^{\square})$. Clearly, at $t^{\square}_{end}$, a path satisfies the query predicate if and only if it is in state ✓: If it is not in state ✓ at time $t^{\square}_{end}$, then it will never reach state ✓, since at any time after $t^{\square}_{end}$, the matrix M^{-} will be used which does not allow to enter state ✓. Thus, the vector $P(t^{\square}_{end})$ has the form $(0, ..., 0, 1)$. Intuitively, this vector corresponds to the assumption, that a path satisfies the query. Now we go back in time using this assumption by using the transposed matrices $(M^{-})^T$ and $(M^{+})^T$: If the current state belongs to $S^{\square}$, we use $(M^{+})^T$, otherwise, we use $(M^{-})^T$. This procedure is repeated until $t = 0$, the last point of time at which an object o has been observed, is reached. The resulting vector v yields, for each state $s \in \mathcal{S}$, the probability that an object starting at s (with a probability of one) satisfies the query. Vector multiplication of the probability distribution $P(o,0)$ with v yields the result probability $P^{\exists}(o, S^{\square}, T^{\square})$.

Example 30. *Again, consider the example depicted in Figure 13.2(c), where we want to compute the probability that an object o intersects the query* $S^{\square} = \{s_1, s_2\}, T^{\square} = \{2, 3\}$. *By transposing* M^{-} *and* M^{+}*, we obtain:*

$$(M^{-})^T = \begin{pmatrix} 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0.8 & 0 \\ 1 & 0.4 & 0.2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and

$$(M^{+})^T = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0 & 0 \\ 1 & 0.4 & 0.2 & 0 \\ 0 & 0.6 & 0.8 & 1 \end{pmatrix}$$

.

The query-based approach starts by assuming the probability distribution $P(t = 3) =$

$(0, 0, 0, 1)$. *Since* $t = 3 \in T^{\square}$, *we compute*

$$P(t = 2) = P(t = 3) \cdot (M^{+})^{T} = (0, 0.6, 0.8, 1).$$

Since $t = 3 \in T^{\square}$ *as well, we repeat this computation:*

$$P(t = 1) = P(t = 2) \cdot (M^{+})^{T} = (0.8, 0.92, 0.96, 1).$$

Finally, since $t = 1 \notin T^{\square}$, *we get*

$$P(t = 0) = P(t = 1) \cdot (M^{-})^{T} = (0.96, 0.864, 0.928, 1)$$

This vector contains, for each state, the probability that an object started at this position will intersect the query. Let us again, assume that initially, the object is located at s_2, *i.e.* $P(o, 0) = (0, 1, 0, 0)$ *we finally obtain:*

$$P^{\exists}(o, S^{\square}, T^{\square}) = P(o, 0) \cdot P(t = 0)^{T} = 0.864.$$

This result equals the result that we derived using the object-based approach.

13.2.3 Discussion

The advantage of the query-based approach is that we only have to compute $P(t = 0)$ once, and then compute, for each object o the $P^{\exists}(o, S^{\square}, T^{\square})$ by one single vector multiplication, which can be performed in $O(|P(o, 0)|)$, where $|P(o, 0)|$ is the number of non-zero elements in $P(o, 0)$, i.e. the number of possible positions of o at $t = 0$. In particular, if we assume that the number of possible states observed at $t = 0$ is small (which is realistic even for inaccurate observation types),we approach a total CPU cost of $O(1)$ per object. The initial computation of $P(t = 0)$ has to perform time-transitions using the transposed Markov-Chain. Thus, a vector-matrix multiplication is required for each transition from $t^{\square}_{end}$ to $t = 0$. Thus, the total runtime of the query-based approach is $O(|\mathcal{DB}| + |S_{reach}|^2 \cdot \delta t)$, where $|\mathcal{DB}|$ is the number of database objects, $|S_{reach}|$ is the number of states that have to be explored, and δt is the number of transitions between $t = 0$ and $t^{\square}_{end}$.

In contrast, the object-based approach has to perform time-transitions using the Markov-Chain *for each object*. Although it is possible in some cases to stop these transitions early using the inherent true-hit detection (computation can be stopped as soon as the probability of state ✓ becomes sufficiently large), in the worst case, all transitions from $t = 0$ to $t^{\square}_{end}$ have to be performed, and for each such transition, a vector-matrix multiplication has to be performed in $O(|S_{reach}|^2)$ time, where $|S_{reach}|$ is the total number of states reachable by o in the time interval $[0, t^{\square}_{end}]$. The total runtime of the object-based approach is thus $O(|\mathcal{DB}| \cdot |S|^2 \cdot \delta t)$.

The query-based approach makes the assumption that all objects follow the same model, i.e., all have the same Markov-Chain. In many applications, this assumption is reasonable: The movements of all icebergs are subject to the same currents - the form and shape of an iceberg can be assumed to have negligible impact on the icebergs movement. In road

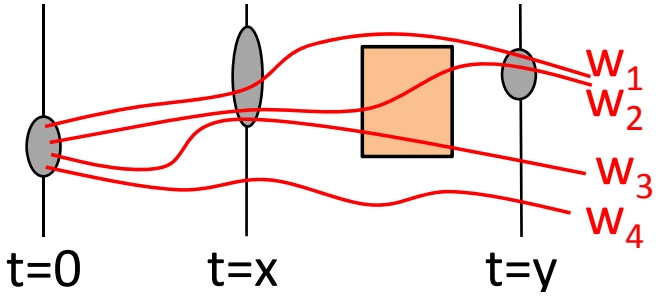


Figure 13.3: Multiple observations of an object

networks, objects may indeed follow different models. In the worst case, we may have to perform the query-based approach once for each object. If the objects can be partitioned to classes (e.g. buses, trucks and cars), the query-based approach can naturally be applied once for each class. In general, if objects follow different Markov-Chains, a technique to speed up the query-based approach is to cluster objects with similar Markov-Chains, and represent each cluster by one approximated Markov-Chain, where each entry is a probability interval instead of a singular probability. This approximated Markov-Chain can be used to perform pruning by detecting clusters of objects which must have (or cannot possibly have) a sufficiently high probability to satisfy the query predicate. Only clusters which cannot be decided as a whole need their objects to be considered individually.

13.3 Multiple Observations

So far, we have assumed that there is only one observation per object and that this observation happened at a time that (temporally) proceeded the query time. In this section, we first show how to compute $P^{\exists}(o, S^{\Box}, T^{\Box})$ given two observations, one before query time and one after query time. Abstractly, this approach can be seen as a time-interpolation, whereas so far we have only considered time-extrapolation. The aim is to incorporate knowledge of both observations, in order to exclude all worlds which are not possible, given both observations, and properly re-weight the probabilities of the remaining worlds. Our proposed technique is applicable for both the object-based as well as the query-based approach. Later, we will give an intuition why considering additional observations, which are further apart from the query window than a considered observation, may still provide additional information. Finally, we outline how these techniques can be easily adapted to incorporate information from an arbitrary number of observations.

Given multiple observations, we have to distinguish between three classes of worlds:

A Worlds that become impossible due to the given observations,

B possible worlds that satisfy the query predicate and

C possible worlds that do not satisfy the query predicate.

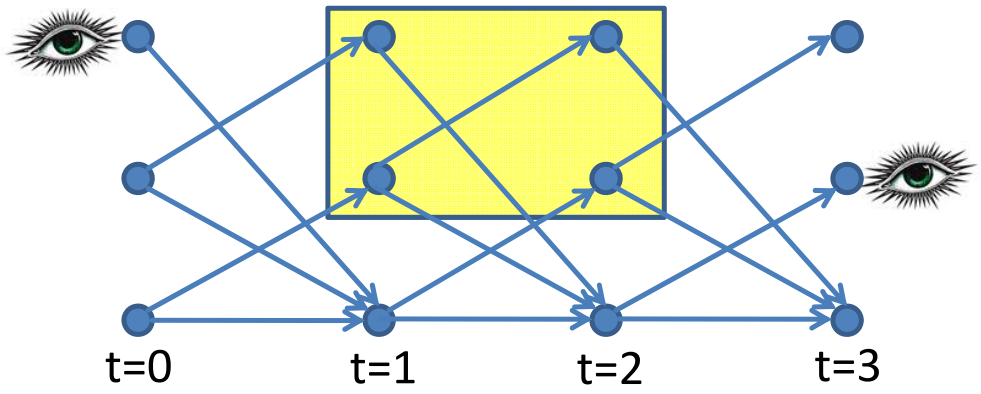


Figure 13.4: Two observations of an object

The example in Fig. 13.3 shows 4 possible worlds (trajectories) of an object o observed at a point of time t_x. In addition, two further observations of o exists at time t_y and t_z. The possible locations of o at each time of the observations are illustrated by the ellipses. Due to the observation made at t_z, worlds w_3 and w_4 become impossible because they include impossible states at time t_z, i.e. both worlds belong to class A. In contrast, w_2 belongs to class B, since this world has a non-zero probability since it includes only possible states at all three observations and satisfies the query predicate of intersecting the query window. Finally, w_1 belong to class C, since, albeit possible, it does not intersect the window.

Intuitively and according to the possible worlds semantics, the probability P_{total} that an object satisfies the query predicate, given some observations, is the fraction of possible worlds that satisfy the query predicate, i.e. the fraction

$$P_{total} = \frac{P(B)}{P(B) + P(C)}. \tag{13.1}$$

In the following, we show how the object-based approach can be adapted to consider multiple observations $OBS^o = \{obs_1^o, \ldots, obs_n^o\}$ of the same object o. Each observation obs_x^o is given by a time $t_{obs_x^o} \in \mathcal{T}$ and a probability distribution $P_{obs_x^o}$ representing the observation. Then, the derived matrices M^- and M^+ can be used for the query-based approach. The approach of Section 13.2.1 cannot be applied directly, because now, worlds which have reached the query window are no longer equivalent, and can no longer be unified in a single state ✓. The reason is that the current state of such a world now effects the probability of reaching the state observed at time $t = 3$. Therefore, we need to maintain, for each world that has intersected the query, information about its current state at each time. Therefore, we replace the state ✓ by a set of states s_1✓,...,s_3✓. Each state s_i✓ corresponds to the probability, that o has intersected the query window *and* is currently located in state s_i. The transition matrices M^- and M^+ need to be adjusted accordingly. The shape of M^- is clear: For a transition from t to $t+1$, where $t+1 \notin T^{\square}$ (the case where M^- is used), states are simply transitioned, and worlds which have (not) intersected the

query at t must (not) have done so at $t+1$. We obtain:

$$M^- = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix}$$

In the case where M^+ is used, that is the case where $t+1 \in T^\square$, we have to ensure that any transition to a state $s \in S^\square$ leads to the corresponding state $s\checkmark$. This yields the matrix

$$M^+ = \begin{pmatrix} M - M' & M' \\ 0 & M \end{pmatrix},$$

where M' is derived from M by setting all columns to zero, where the corresponding state $s \notin S^\square$, and $M - M'$ is derived by setting all columns to zero for which $s \in S^\square$.

We start by using the probability distribution $P_{obs_1^o}$ of an object o observed at $t_{obs_1^o}$. As in the previous sections, we iteratively apply the modified matrices M^- and M^+ until we reach $t_{obs_2^o}$. At this point, we have two probability distributions: One distribution derived using the observation obs_1^o, which has been transitioned to $t_{obs_2^o}$, as well as the probability distribution $P_{obs_2^o}$. These observations can be unified by exploiting independence between observations.

Lemma 37. *Let $X(t) := \{P_{obs_1^o}(t), \ldots, P_{obs_n^o}(t)\}$ be a set of pdfs of an object o at time t, derived from independent observations. The joint probability distribution $P(o,t)$ of o at time t is given by:*

$$P(o,t) = N(\prod_{x \in X} x_1, ..., \prod_{x \in X} x_{|\mathcal{S}|}),$$

where $N(\cdot)$ is the vector normalization function, i.e. $N(x) = (\frac{x_1}{\sum x}, ..., \frac{x_{|\mathcal{S}|}}{\sum x})$

Proof. For each $i \in 1, ..., |\mathcal{S}|$, $P(o,t)_i$ is, by definition of a probability density function, the probability of the random event that object o is located in state s_i. Without loss of generality, let $a = (a_1, ..., a_{|\mathcal{S}|}) \in X$ be the first observation. Given this observation only, then clearly $P(o,t) = a$ holds. Given further observations, the probability of this event becomes conditioned to

$$P(o,t)_i = P(a_i | X \setminus \{a\}).$$

Since all observations are mutually independent, we get

$$P(o,t)_i = a_i \cdot \prod_{x \in X} x_i,$$

which is the fraction of all worlds, including worlds which are no longer possible given the observations X, in which o is located in state s_i at time t. Due to possible worlds semantics (c.f. Equation 13.1), we are only interested in the fraction of possible worlds in which o is located in state s_i at time t. Since v contains all possible worlds, the normalization $N(v)$ yields the correct result. □

As an example, consider Figure 13.4, where we again use our running example Markov-Chain

$$M = \begin{pmatrix} 0 & 0 & 1 \\ 0.5 & 0 & 0.5 \\ 0 & 0.8 & 0.2 \end{pmatrix}$$

and assume that an object O has been observed at state s_1 at time t_1 and at state s_3 at time t_4. We obtain the transition matrices

$$M^- = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0.5 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.8 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

and

$$M^+ = \begin{pmatrix} M - M' & M' \\ 0 & M \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0.5 & 0.5 & 0 & 0 \\ 0 & 0 & 0.2 & 0 & 0.8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0.5 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

Since at time $t = 0$, o has been observed in state s_1, and since o cannot have reached the window yet, the initial distribution of o is $P_{obs_1^o} = P(o, 0) = (1, 0, 0, 0, 0, 0)$. Transition to $t = 1$ using M^+ (since $t = 1 \in T^\square$) yields $P(o, 1) = (0, 0, 1, 0, 0, 0)$. The next transition using M^+ yields $P(o, 2) = (0, 0, 0.2, 0, 0.8, 0)$. The intuition of this vector is that, at time $t = 2$, object o is located in state s_3 while having reached the query window with a probability of 20%, and otherwise is located in state s_2 having reached the query window. The next transition uses M^- (since $t = 3 \notin T^\square$) and yields $P(o, 3) = (0, 0.16, 0.04, 0.4, 0, 0.4)$. Now, at time $t = 3$, the second observation was made. We assume that this observation only has information about the state at $t = 3$, but no information whether o has intersected the query window. Thus, the observation vector has the form $obs_2^o = (0, 0.5, 0, 0, 0.5, 0)$. Due to Lemma 37, and since we assume that obs_1^o (from which $P(o, 3)$ was derived) and obs_2^o are independent observations, we can directly multiply the entries of $P(o, 3)$ and obs_2^o, yielding $P(o, t)\prime = (0, 0.08, 0, 0, 0, 0)$. Normalization yields $P(o, t) = N(P(o, t)\prime) = (0, 1, 0, 0, 0, 0)$, which means that at $t = 3$, given both observations, o must be in state s_2 and must not have intersected the query window. This is intuitive, since the only path between s_1 at $t = 0$ and s_2 at $t = 3$ does not intersect the query window.

13.4 Additional Spatio-Temporal Queries

Based on the proposed concepts for answering spatio-temporal ∃-queries, we will now show how different query predicates can be answered efficiently.

PST∀Q:

In some applications, it may be interesting to compute the probability that o is in the query window at *all* times $t \in T^{Box}$. Clearly, the probability that o is located in $S^\square$ at all times in $T^\square$ complements the probability that o is outside $S^\square$ at any time in $T^\square$, i.e.

$$P^\forall(o, S^\square, T^\square) = 1 - P^\exists(o, \mathcal{S} \setminus S^\square, T^\square).$$

Although, in general, $\mathcal{S} >> S^\square$, the time required to compute $P^\exists(o, \mathcal{S} \setminus S^\square, T^\square)$ is generally not larger than the time required to compute $P^\forall(o, S^\square, T^\square)$. The only difference between these two computations is the content of the matrix M^+, since different sets of columns of matrix M are merged. In most cases, the computation of $P^\exists(o, \mathcal{S} \setminus S^\square, T^\square)$ is actually faster, since more columns of M^+ are zero.

PSTkQ:

So far, an object o satisfies the query predicate in any world where it intersects the query window, regardless for how long o remains in the query window. In the following, we will show how the probability distribution of the number of times that o is located in the query window (c.f. Definition 63) is computed.

To answer this query, we can extend the idea of adding new virtual states, to capture the number of times an object has visited the query window; we use the set of states $\mathcal{S}' = \mathcal{S} \times \{0, ..., |T^\square|\}$. Intuitively, an object in state $s' = (s \in \mathcal{S}, k \in \{0, ..., |T^\square|\}) \in \mathcal{S}'$ is currently located in state s and has been in the query window at k points of time. At any point of time $t \in T^\square$, all possible worlds located at a state $s \in S^\square$ are transitioned in order to increase their k value by one. The resulting matrices are

$$M^- = \begin{pmatrix} M & 0 & ... & 0 \\ 0 & M & ... & 0 \\ ... & ... & ... & ... \\ 0 & 0 & ... & M \end{pmatrix}$$

$$M^+ = \begin{pmatrix} M - M' & M' & 0 & ... & 0 \\ 0 & M - M' & M' & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & ... & M - M' & M' \end{pmatrix}$$

Since initially, an object o visited the query window zero times[3], the initial distribution of an object is concatenated with $k \cdot |\mathcal{S}|$ zeroes. If we perform time transitions until we reach

[3]The special case where $t = 0$ belongs to $\mathcal{T}^\square$ is omitted here. In that case, the initial distribution of an object simply starts at $k = 1$, i.e. is shifted by $|\mathcal{S}|$.

$t^{\square}_{end}$, we obtain for each $s' = (s \in \mathcal{S}, k \in \{0, ..., |T^{\square}|\}) \in \mathcal{S}'$ the probability that o will visit the query window exactly k times and will finally be in state s. Grouping this result by k, we obtain for each $k \in \{0, ..., |T^{\square}|\}$, the probability that o visits the query window exactly k times.

Obviously the above approach is not very memory efficient, since M^- and M^+ each blow up the memory requirement of the original transition matrix M by a factor of $|T^{\square}|$. Thus, a more space efficient approach is presented in the following. For processing an object $o \in \mathcal{DB}$, an additional $(|T^{\square}| + 1) \times |\mathcal{S}|$-Matrix $C(t)$ is necessary. Each entry $c_{i,j}(t) \in C(t)$ corresponds to the probability that o is currently located in state s_j and has been located in $S^{\square}$ at exactly i points of time $t' \leq t \in T^{\square}$. We begin by setting the first row of $C(0)$ to $P(o, 0)$ and all other entries to zero, since at $t = 0$, o cannot possibly have entered the query region. At each state transition from t to $t + 1$, a transition using M is performed for each row. This is simply achieved by computing $C'(t + 1) = C(t) \cdot M$. If t is not in $T^{\square}$, then we set $C(t+1) = C'(t+1)$. Otherwise, if $t \in T^{\square}$, then we additionally shift each column corresponding to a state $s_i \in \mathcal{S}$ down by one, and fill the top entry of this line with zero. That is

$$c_{i,j} = \begin{cases} c'_{i,j}, & \text{if } j \notin S^{\square} \\ 0, & \text{if } j \in S^{\square} \wedge i = 0 \\ c'_{i-1,j} & \text{otherwise.} \end{cases}$$

When $t^{\square}_{end}$ is reached, the probability for o to be in the query exactly k-times can be obtained by summing up all probabilities in row k of $C(t^{\square}_{end})$. Thus the probability $P^{k-times}(o, S^{\square}, T^{\square})$ that o has visited the query window exactly k times is given by

$$P^{k-times}(o, S^{\square}, T^{\square}) = \sum_j c_{k,j}(t^{\square}_{end})$$

Considering our running example we start with the matrix $C(0)$ and transition as along as $t \notin T^{\square}$:

$$C(0) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot M^2} C(2)' = \begin{pmatrix} 0 & 0.32 & 0.68 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Now we shift down the probabilities of the states in $S^{\square}$ by one row:

$$C(2) = \begin{pmatrix} 0 & 0 & 0.68 \\ 0 & 0.32 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

A further transition using the Markov chain yields

$$\begin{pmatrix} 0 & 0 & 0.68 \\ 0 & 0.32 & 0 \\ 0 & 0 & 0 \end{pmatrix} \xrightarrow{\cdot M} C(3)' = \begin{pmatrix} 0 & 0.544 & 0.136 \\ 0.192 & 0 & 0.128 \\ 0 & 0 & 0 \end{pmatrix}$$

After performing the last shift we obtain

$$C(3) = \begin{pmatrix} 0 & 0 & 0.136 \\ 0 & 0.544 & 0.128 \\ 0.192 & 0 & 0 \end{pmatrix} \xrightarrow{rowsum} \begin{pmatrix} 0.136 \\ 0.672 \\ 0.192 \end{pmatrix}$$

The resulting vector reflects the probability of o to be in the query exactly zero times ($P^{0-times}(o, S^{\square}, T^{\square}) = 0.136$), exactly once ($P^{1-times}(o, S^{\square}, T^{\square}) = 0.672$) and exactly two times ($P^{2-times}(o, S^{\square}, T^{\square}) = 0.192$).

13.5 Conclusion

In this chapter, we studied the problem of probabilistic query evaluation over uncertain spatio-temporal data. We consider uncertain trajectories, for which some points are sampled via observations, while the remaining points are instantiated by a stochastic process. To our knowledge, this is the first work that studies such queries over uncertain moving object data, which are modeled by stochastic processes, specifically Markov-Chains. This approach has three major advantages over previous work. First it allows answering queries in accordance with the possible worlds model. Second, dependencies between object locations at consecutive points in time are taken into account. And third, it allows us to infer the probability an object reaches a certain location (state) by matrix multiplications that can be processed extremely efficient. Based on this method we propose a framework for processing queries over such data, which injects pruning techniques into the Markov Chain matrices. An object-based and a query-based approach are proposed; the latter is always more efficient, typically by orders of magnitude. In our experiments we show that we are able to answer queries on settings with 100,000 location states and 10,000 objects in a fraction of a second on a single machine in contrast to state-of-the-art solutions that are multiple orders of magnitude slower, e.g. the Monte-Carlo approach requires several hours for the same query. We show how the framework can be applied for the cases where there exist one or multiple observations per object and for various probabilistic spatio-temporal query variants.

We believe that many more interesting queries and applications can be set on top of this model. To support this we release the MATLAB framework which was developed during the process of this work online[4]. In the future, we plan to apply our framework for data analysis tasks over spatio-temporal data (e.g. find areas that are expected to become congested together with the time periods of this expectation).

[4]The project page can be found at http://www.dbs.ifi.lmu.de/cms/Publications/UncertainSpatioTemporal

Chapter 14

Spatio-Temporal Nearest Neighbor Queries

Given a reference state or trajectory q and a time interval T, we define *probabilistic* nearest-neighbor (PNN) query semantics, which are extensions of nearest neighbor queries in trajectory databases [72, 80, 96, 182]. Specifically, a P$\exists$NNQ (P$\forall$NNQ) query retrieves all objects in $\mathcal{DB}$, which have sufficiently high probability to be the NN of q at one time (at the entire set of times) in T; a probabilistic *continuous* NN (PCNNQ) query finds for each object $o \in \mathcal{DB}$ the time subsets T_i of T, wherein o has high enough probability to be the NN of q at the entire set of times in T_i. Note that to the best of our knowledge this is the first approach that tackles the PNN query problem correctly in consideration of possible worlds semantics.

PNN queries find several applications in analyzing historical trajectory data. For example, consider a geo-social network where users can publish their current spatial position at any time by so-called check-ins. For a historical event, users might want to find their nearest friends during this event, e.g. to share pictures and experiences. As another application example, consider GPS-tracked taxi cars as given in the T-Drive dataset [209] where PNN queries can be used for analysis tasks like the assessment of taxi-client assignment procedures or for search tasks like searching for taxi drivers that might have observed a certain event like a car accident or a criminal activity such as a bank robbery. The taxi drivers that have been closest to the certain event location during the time the event might happened are potential witnesses. Note that this example application is used as our running application throughout this paper.

The main contributions of our work are as follows:

- A thorough theoretical complexity analysis for variants of probabilistic NN query problems.
- A sampling-based approximate solution applicable for all PNN problems which is based on Bayesian inference.

The rest of the chapter is structured as follows. Section 14.1 reviews existing work

related to NN search on uncertain trajectories. Section 14.2 presents variants of nearest neighbor search semantics on spatio-temporal data. Section 14.3 gives a theoretical analysis for each variant. To speed up query evaluation, in Chapter 15, we show that it is possible to prune some objects from consideration using an index over $\mathcal{DB}$. Then, for each remaining object o, we have to compute a probability (i.e., $P\exists NN(o, q, \mathcal{DB}, T)$ or $P\forall NN(o, q, \mathcal{DB}, T)$) and compare it to the threshold τ. In Section 14.3, it is shown that the problems of answering P∃NN queries, P∀NN queries and PCNN queries is computationally hard. To alleviate this problem, Chapter 16 present approximate solutions for this kind of spatio-temporal queries using Monte-Carlo simulation. A thorough experimental analysis presented in Chapter 17 shows that this sampling approach yields low run-times by exploiting efficient matrix operations, and yields result probabilities having an error that can practically be neglected.

14.1 Related Work

In the context of certain trajectory databases there is not a common definition of nearest neighbor queries, but rather a set of different interpretations. In [72], given a query trajectory (or spatial point) q and a time interval T, a NN query returns either the trajectory from the database which is closest to q during T or for each $t \in T$ the trajectory which is closest to q. The latter problem has also been addressed in [80]. Similarly, in [103], all trajectories which are nearest neighbors to q for at least one point of time t are computed.

Other approaches consider *continuous* nearest neighbor (CNN) semantics. In [96], CNN queries were defined taking as input a static spatial query point q and a trajectory database and returning for each point in time the trajectory closest to q. Other approaches [153, 182] define the CNN problem differently: Given an input trajectory q and a database consisting of spatial points, a CNN query segments q such that for each segment $q_i \subseteq q$ exactly one object from the database is the nearest neighbor of q_i. This approach was extended for objects with uncertain velocity and direction, thus considering a predictive setting rather than historical data, in [94]; the solutions proposed only find possible results, but not result probabilities. Solutions for road network data were also proposed for the case where the velocities of objects are unknown [122]. Furthermore, [190, 189] extended the problem of continuous kNN queries (on historical search) to an uncertain setting, serving as important preliminary work, however, based on a model which is not capable to return answers according to possible world semantics.

14.2 Problem Definition

In this chapter, we consider three types of time-parameterized nearest-neighbor queries that take as input a certain trajectory q and a set of timesteps T.

Definition 64 (P∃NN Query)**.** *A probabilistic ∃ nearest neighbor query retrieves all objects $o \in \mathcal{DB}$ which have a sufficiently high probability ($P\exists NN$) to be the nearest neighbor of q*

for at least one point of time $t \in T$, formally:

$$P\exists NNQ(q, \mathcal{DB}, T, \tau) = \{o \in \mathcal{DB} : P\exists NN(o, q, \mathcal{DB}, T) \geq \tau\}$$
$$\text{where } P\exists NN(o, q, \mathcal{DB}, T) =$$
$$P(\exists t \in T : \forall o' \in \mathcal{DB} \setminus o : d(q(t), o(t)) \leq d(q(t), o'(t)))$$

and $d(x, y)$ is a distance function defined on spatial points, typically the Euclidean distance.

This definition is a straightforward extension of the spatio-temporal query proposed in [72]. In addition, we consider NN queries with the $\forall$ quantifier which has been introduced in the previous chapter for spatio-temporal window queries).

Definition 65 (P$\forall$NN Query)**.** *A probabilistic $\forall$ nearest neighbor query retrieves all objects $o \in \mathcal{DB}$ which have a sufficiently high probability ($P\forall NN$) to be the nearest neighbor of q for the entire set of timestamps T, formally:*

$$P\forall NNQ(q, \mathcal{DB}, T, \tau) = \{o \in \mathcal{DB} : P\forall NN(o, q, \mathcal{DB}, T) \geq \tau\}$$
$$\text{where } P\forall NN(o, q, \mathcal{DB}, T) =$$
$$P(\forall t \in T : \forall o' \in \mathcal{DB} \setminus o : d(q(t), o(t)) \leq d(q(t), o'(t)))$$

In addition to the $\exists$ and $\forall$ semantics for probabilistic nearest neighbor queries we now introduce a *continuous* query type which intuitively extends the spatio-temporal continuous nearest-neighbor query [153, 182] to apply on uncertain trajectories.

Definition 66 (PCNN Query)**.** *A probabilistic continuous nearest neighbor query retrieves all objects $o \in \mathcal{DB}$ together with the set of timesets $\{T_i\}$ where in each T_i the object has a sufficiently high probability to be always the nearest neighbor of $q(t)$, formally:*

$$PCNNQ(q, \mathcal{DB}, T, \tau) =$$
$$\{(o, T_i) : o \in \mathcal{DB}, T_i \subseteq T, P\forall NN(o, q, \mathcal{DB}, T_i) \geq \tau\}.$$

Analogously to the CNN query definition [153, 182], in order to reduce redundant answers it makes sense to redefine the PCNN Query where we focus on results that maximize $|T_i|$, formally:

$$PCNNQ(q, \mathcal{DB}, T, \tau) =$$
$$\{(o, T_i) : o \in \mathcal{DB}, T_i \subseteq T, P\forall NN(o, q, \mathcal{DB}, T_i) \geq \tau$$
$$\wedge \forall T_j \supset T_i : P\forall NN(o, q, \mathcal{DB}, T_i) < \tau\}.$$

Note that according to this definition result sets $T_i \subseteq T$ do not have to be connected.

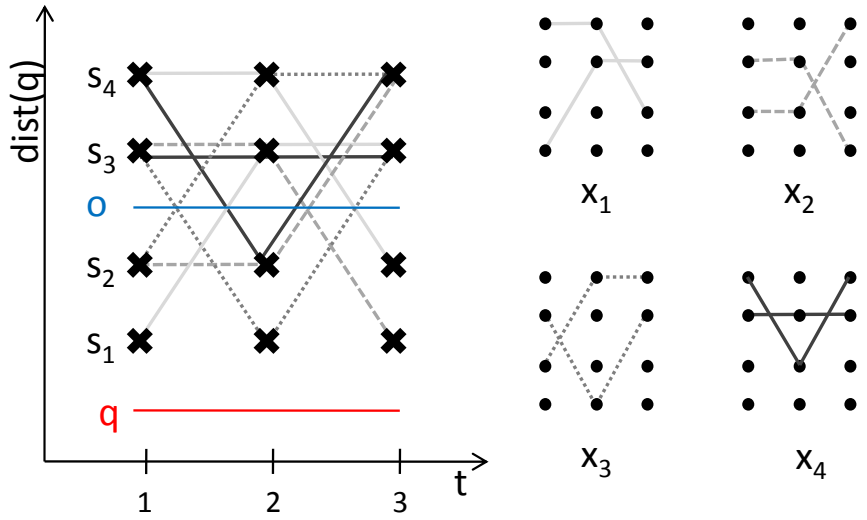


Figure 14.1: An example instance of our mapping of the 3-SAT problem to Markov chains.

14.3 Theoretical Analysis

In this section, we formally show that P∃NN queries, P∀NN queries and and PCNN queries cannot be answered in PTIME..

14.3.1 The P∃NN Query

In a P∃NN query, for any candidate object $o \in \mathcal{DB}$, we should consider the probability $P\exists NN(o, q, \mathcal{DB}, T)$. However, the following lemma shows that this probability is hard to compute.

Lemma 38. *The computation of* $P\exists NN(o, q, \mathcal{DB}, T)$ *is NP-hard in* $|\mathcal{DB}|$.

Proof. $P\exists NN(o, q, \mathcal{DB}, T)$ is equal to $1-P(\neg\exists t \in T, \forall o' \in \mathcal{DB} : d(q(t), o(t)) \leq d(q(t), o'(t)))$. We will show that deciding if there exists a possible world for which the expression:

$$\neg\exists t \in T, \forall o' \in \mathcal{DB} : d(q(t), o(t)) \leq d(q(t), o'(t)) \tag{14.1}$$

is satisfied is an NP-hard problem. (Note that this is a much easier problem than computing the actual probability.) Specifically, we will reduce the well-known NP-hard k-SAT problem to the problem of deciding on the existence of a possible world for which Expression 14.1 holds.

For this purpose, we provide a mapping to convert a boolean formula in conjunctive normal form to a Markov chain modeling the decision problem of Expression 14.1 in polynomial time. Thus, if the decision problem could be computed in PTIME, then k-SAT could also be solved in PTIME, which would only be possible if P=NP. A k-SAT expression E is based on a set of variables $X = \{x_1, x_2, \ldots, x_n\}$. The *literal* l_i of a variable x_i is either

x_i or $\neg x_i$ and a *clause* $c = \bigvee_{x_i \in \mathbb{C}} l_i$ is a disjunction of literals where $\mathbb{C} \subseteq X$ and $|\mathbb{C}| < k$. Then E is a conjunction of clauses: $E = c_1 \wedge c_2 \wedge \ldots \wedge c_m$.

For our mapping, we will consider a simplified version of the P∃NN problem, specifically (1) q is a certain point, (2) o is a certain point and (3) the state space $\mathcal{S}$ of possible locations only includes 4 states. As illustrated in Figure 14.1, compared to o, states s_1 and s_2 are closer to q and states s_3 and s_4 are further from q.[1] Therefore, if an uncertain object is at states s_1 or s_2 then o is not the NN of q.

In our mapping, each variable $x_i \in X$ is equivalent to one uncertain object $o'_i \in \mathcal{DB} \setminus o$. Furthermore each disjunctive clause c_j is interpreted as an event happening at time $t = j$, i.e., the event c_1 happens at time $t = 1$, c_2 happens at time $t = 2$ etc. Each clause c_j can be seen as a disjunctive event that at least one object o'_i at time $t = j$ is closer to q than o (in this case, c_j is *true*). Therefore, the conjunction of all these events, i.e. expression $E = \bigwedge_{1 \leq j \leq m} c_j$, becomes true if the set of variables is chosen in a way that at each point in time, compared to o, at least one object is closer to q; this directly represents Expression 14.1. However, in k-SAT, not every variable x_i (corresponding to o'_i) is contained in each term c_j which does not correspond to our setting, since an uncertain object has to be *somewhere* at each point in time. To solve this problem, we extend each clause c_j, such that each variable x_i is contained in c_j, without varying the semantics of c_j. Let us assume that x_i is not contained in c_j. Then $c'_j = c_j \vee false = c_j \vee (x_i \wedge \neg x_i)$. This means that we can assume that object o'_i is definitely not closer to q than o at time t.

Let l_i^j be the literal of variable x_i in clause c_j. Based on the above discussion, we are able to construct for each object o'_i two possible trajectories (worlds). The first one, based on the assumption that x_i is true, transitions between states s_2 (if l_i^j = true) and s_4 (if l_i^j = false). The second one, based on the assumption that x_i is set to false, transitions between states s_1 (if l_i^j = true) and s_3 (if l_i^j = false). Since these two trajectories can never be in the same state it is straightforward to construct a time-inhomogeneous Markov chain $T^o(t)$ for each object o'_i and each timestamp j.

After the Markov chains for each uncertain object o'_i in $\mathcal{DB}$ have been determined, we would just have to traverse them and compute the probability $P\exists NN(o, q, \mathcal{DB}, T)$. If this probability is < 1, there would exist a solution to the corresponding k-SAT formula. However it is not possible to achieve this efficiently in the general case as long as $P \neq NP$. Therefore solving $P\exists NN$ in subexponential time is impossible. □

Example: Consider a set of boolean variables $X = \{x_1, \ldots, x_4\}$ and the following formula:

$$E = (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee \neg x_2)$$

Therefore, we have

$$c_1 = (\neg x_1 \vee x_2 \vee x_3), c_2 = (x_2 \vee \neg x_3 \vee x_4) \text{ and } c_3 = (x_1 \vee \neg x_2)$$

[1] The states of o and q are omitted for the sake of simplicity.

By employing the mapping discussed above, we get the four inhomogeneous Markov chains illustrated in Figure 14.1. For instance, under the condition that x_1 is set to $true$, the value of the literal $\neg x_1$ is false at $t = 1$ (in clause c_1) such that o'_1 starts in the state s_4. On the other hand, if x_1 is set to $false$, then o'_1 starts in the state s_1.

In the second clause c_2, since $x_1 \notin \mathbb{C}_2$, the position of o'_1 must not affect the result. Therefore, for both cases $x_1 = false$ and $x_1 = true$, o'_1 must be behind o. In the last clause c_3, if $x_1 = true$ the object moves to state s_2. On the other hand, if $x_1 = false$, the object moves to state s_3.

14.3.2 The P∀NN Query

In the following, let $o \prec_q^T o_a$ denote the random predicate that is true if and only if object o is closer to an object q than object $o_a \in \mathcal{DB}$ during the query time $T = [t_{start}, t_{end}]$, i.e. $\forall t \in T : d(o(t), q(t)) \leq d(o_a(t), q(t))$. If $o \prec_q^T o_a$ holds, we say that o *dominates* o_a with respect to q during T. If the parameters q and T are clear from the context, we simply say that o dominates o_a. By definition, an object o is a ∀-nearest neighbor of object q if an only if o dominates all other objects in $\mathcal{DB}$. The following theoretically analysis of the P∀NN query is organized as follows:

- It will be shown in Lemma 39 that the probability that o dominates a single object o_a can be computed in PTIME. This is an interesting theoretical result, since we have seen in Section 14.3.1 that this is not possible for P∃NN queries.

- Despite Lemma 39, it will be shown in Lemma 40 that computing the probability $P\forall NN(o, q, \mathcal{DB}, T)$ that an object o is a forall-nearest neighbor of q is hard to compute, due to stochastic dependencies between domination random events $o \prec_q^T o_a$ and $o \prec_q^T o_b$.

Lemma 39. *The probability $P(o \prec_q^T o_a)$ that o dominates o_a can be computed in PTIME.*

Proof. The main idea of this proof, is to treat the positions of o and o' as a single joint stochastic process, having possible alternatives in $\mathcal{S}^2$. Then, the joint a-priori transition matrix $M_{o\times o'}(t)$ will be conditioned to the event $o \prec_q^T o'$ following the forward-backward paradigm. For completeness, this paradigm will be presented in detail as part of the following proof.

Let $s_{o\times o'}(t)$ denote the joint probability distribution of o and o' at time t, conditioned to the event $o \prec_q^{[t_0,t-1]} o'$ that o dominates o' at any time prior to t. For ease of implementation, we linearize this two dimensional discrete distribution $s_{o\times o'}(t)$ to a vector of length $|\mathcal{S}|^2$, such that each entry $(s_{o\times o'}(t))_i$ equals the probability that, at time t, object o is in state i mod $|\mathcal{S}|$ and object o' is in state $idiv|\mathcal{S}|$, given that o dominates o' at any time prior to t. Here, the operator div is the integer division operator, and mod is the remainder of the integer division. Formally, each entry of vector o dominates o' at any time prior to t For instance, if we have a total of 1000 states, vector $s_{o\times o'}(t)$ has a length of one million entries,

and entry $i = 484.912$ contains the probability that o is located at state 912 at time t and o' is located at state 484 at time t. Formally, each entry of $s_{o\times o'}(t)$ is defined as:

$$s_{o\times o'}(t))_i := P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} | o \prec_q^{[t_0,t-1]} o'.$$

Furthermore, let $M_{o\times o'}(t)$ denote the joint transition matrix of o and o'. This matrix contains the probabilities of objects o and o' to transition to a joint state j from a joint state i at time t. That is, $M_{o\times o'}(t)$ is a $|\mathcal{S}|^2 \times |\mathcal{S}|^2$ matrix, such that each entry $(M_{o\times o'}(t))_{i,j}$ corresponds to the probability

$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t+1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|}).$$

Due to independence of o and o', these probabilities can be computed by

$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t+1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|}) =$$

$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|}) \cdot$$
$$P(o'(t+1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|}) =$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|}) \cdot P(o'(t+1) = s_{j\ div\ |\mathcal{S}|} | o'(t) = s_{i\ div\ |\mathcal{S}|}) =$$
$$= (M_o(t))_{i \bmod |\mathcal{S}|, j \bmod |\mathcal{S}|} \cdot (M_{o'}(t)_{i\ div\ |\mathcal{S}|, j\ div\ |\mathcal{S}|}).$$

In the forward phase, the idea is to start at time t_0, using the joint distribution $s_{o\times o'}(t_0)$, removing worlds that contradict the predicate $o \prec_q^T o'$ and iteratively performing a transition to $s_{o\times o'}(t+1)$ given $s_{o\times o'}(t)$. In each iteration, i.e., at each point of time in the query interval T, Bayesian inference is used to construct a time-reversed Markov-model $R_{o\times o'}(t)$ of o and o' at time t given the observation that o has dominated o' in the past, i.e., a model that describes the probabilities

$$R_{o\times o'}(t)(i,j) =$$
$$P(o(t-1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t-1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o')$$

of o and o' coming from a state pair j at time $t-1$, given being at a state pair i at time t, given the observation that o has dominated o' in the past.

In the Backward-phase, time is traversed in reverse direction, from time t_{end} to t_0, by employing the time-reversed Markov-model $Ro \times o'(t)$ constructed in the forward phase. Again, Bayesian inference is used to construct a new Markov model $Fo \times o'(t)$ that is further adapted, given the observation that o will dominate o' in the future. This new Markov model contains the final a-posteriori transition probabilities

$$F_{o\times o'}(t)(i,j) =$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t+1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} \wedge o \prec_q^T o').$$

In more detail, the forward phase uses the theorem of Bayes to compute backward probabilities as follows

$$R_{o\times o'}(t)(i,j) =$$
$$P(o(t-1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t-1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o') =$$
$$P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} |$$
$$o(t-1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t-1) = s_{j\ div\ |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o') \cdot$$

$$\frac{P(o(t-1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t-1) = s_{j\ div\ |\mathcal{S}|}) | o \prec_q^{[t_0,t]} o'}{P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} | o \prec_q^{[t_0,t]} o')} \tag{14.2}$$

Equation 14.2 requires two classes of probabilities to be computed on its right-hand-side:

- The a-priori probabilities

$$P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} |$$
$$o(t-1) = s_{j \bmod |\mathcal{S}|} \wedge o'(t-1) = s_{j\ div\ |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o'),$$
$$1 \leq i,j \leq |\mathcal{S}|^2, t_0 \leq t \leq t_{end}$$

  of a joint transition of o and o' at time t, given that o has dominated o' at any time less or equal t, and

- The priors $s_{o\times o'}(t_0))_i := P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i\ div\ |\mathcal{S}|} | o \prec_q^{[t_0,t]} o'), 1 \leq i,j \leq |\mathcal{S}|^2, t_0 \leq t \leq t_{end}$

The a-priori probabilities are given directly by the joint a-priori Markov chain $M_{o\times o'}(t)$. Exploiting the Markov property of the transition matrix $M_{o\times o'}(t)$, we can rewritte each element of $M_{o\times o'}(t)$ as

$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j\ div\ |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i\ div\ |\mathcal{S}|}) =$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j\ div\ |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i\ div\ |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o').$$

Due to the Markov assumption, the future positions of o and o' are independent of past knowledge $o \prec_q^{[t_0,t-1]} o'$. Due to precise knowledge of the position of o and o' at time t, the future positions are independent of the current knowledge $o \prec_q^{[t,t]} o'$. Thus, the future positions of o and o' are independent of $o \prec_q^{[t_0,t]} o'$. Consequently, the probability $P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j\ div\ |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i\ div\ |\mathcal{S}|}$, which corresponds to the probability of future positions of o and o' given current positions of o and o' is unaffected by conditioning to $o \prec_q^{[t_0,t]} o'$.

These prior probabilities are computed inductively as follows. At time t_0, we can exploit the following simplification of vector $s_{o\times o'}(t_0)$

$$(s_{o\times o'}(t_0))_i = P(o(t_0) = s_{i \bmod |\mathcal{S}|} \wedge o'(t_0) = s_{i \; div \; |\mathcal{S}|} | o \prec_q^{[t_0,t_0-1]} o') =$$

$$(s_{o\times o'}(t_0))_i := P(o(t_0) = s_{i \bmod |\mathcal{S}|} \wedge o'(t_0) = s_{i \; div \; |\mathcal{S}|}),$$

which follows from the fact that the predicate $o \prec_q^{[t_0,t_0-1]} o'$ is a tautology, as it is defined on an empty time interval $[t_0, t_0 - 1]$. This observation follows from the fact that $o \prec_q^{[t_0,t_0-1]} o'$ is defined as $\forall t_0 \leq t \leq t_0 - 1 : dist(q(t), o(t)) \leq dist(q(t), q'(t))$ which simplifies to $\forall t \in \emptyset \phi$ which is a tautology for any formula ϕ. Due to independence of o and o', we can compute the initial probabilities

$$P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o'(t) = s_{i \; div \; |\mathcal{S}|}) =$$

$$P(o(t) = s_{i \bmod |\mathcal{S}|}) \cdot P(o'(t) = s_{i \; div \; |\mathcal{S}|}).$$

Next, all entries of $(s_{o\times o'}(t_0))_i$ are set to zero, for which it holds that the distance of the corresponding state of o has a higher distance to the query trajectory q than the corresponding state of o'. That is, each entry $(s_{o\times o'}(t_0))_i$ such that $dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|})$ is set to zero. This operation can be defined via matrix-operations only by defining the following indicator vector:

$$C_i(t) = \begin{cases} 1, & \text{if } dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|}) \\ 0, & \text{otherwise.} \end{cases}$$

Semantically, the indicator vector $C(t)$ is used to prune all worlds, for which it holds that $dist(q(t), o(t)) > dist(q(t), o'(t))$ at time t. Element wise multiplication yields a new vector

$$s_{o\times o'}(t_0)' := s_{o\times o'}(t_0) \bullet C(t_0).$$

By definition each cell $C(t_0)_i$ equals the probability $dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|})$. This probability must be either one or zero, since it is completely deterministic, given the position of q, o and o' at time t. Thus we can rewrite each element of $s_{o\times o'}(t_0)' := s_{o\times o'}(t_0) \bullet C(t_0)$ as follows

$$(s_{o\times o'}(t_0)')_i =$$

$$P(o(t_0) = s_{i \bmod |\mathcal{S}|} \wedge o'(t_0) = s_{i \; div \; |\mathcal{S}|}) \cdot$$

$$P(dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|})).$$

Since the event $dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|})$ is deterministic, it is in particular independent of any other random variable, such that we can write

$$(s_{o\times o'}(t_0)')_i =$$

$$P(o(t_0) = s_{i \bmod |\mathcal{S}|} \wedge o'(t_0) = s_{i \; div \; |\mathcal{S}|}$$
$$\wedge dist(q(t), s_{i \bmod |\mathcal{S}|}) > dist(q(t), s_{i \; div \; |\mathcal{S}|})).$$

By definition of $o \prec_q^{t_0} o'$, we can write

$$(s_{o\times o'}(t_0)')_i = P(o(t_0) = s_{i \bmod |\mathcal{S}|} \wedge o'(t_0) = s_{i \; div \; |\mathcal{S}|} \wedge o \prec_q^{t_0} o').$$

Using the joint transition matrix $M_{o\times o'}(t)$ we can now perform a time transition from time t_0 to time $t_0 + 1$ by exploiting the following equation.

$$s_{o\times o'}(t+1) = M_{o\times o'}(t) \cdot s_{o\times o'}(t)'. \tag{14.3}$$

Correctness of Equation 14.3 can be shown using the law of total probability: According to Equation 14.3, each entry $(s_{o\times o'}(t+1))_j$ is computed by the sum

$$\sum_i (s_{o\times o'}(t))_i \cdot (M_{o\times o'}(t))_{i,j} =$$

$$\sum_i P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|} | o \prec_q^{[t_0,t]} o') \cdot$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|})$$

Exploiting the Markov property of the transition matrix $M_{o\times o'}(t)$, we can rewritte each element of $M_{o\times o'}(t)$ as

$$P(o(t+1) = o(t) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|}) =$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o').$$

The addition of the condition $o \prec_q^{[t_0,t]} o'$ has no effect, since these probabilities are already conditioned to positions $o(t)$ and $o'(t)$. Due to the Markov assumption, the future positions of o and o' are independent of past knowledge $o \prec_q^{[t_0,t-1]} o'$, and due to precise knowledge of the position of o and o', the future positions are independent of the current knowledge $o \prec_q^{[t,t]} o'$. Thus, the future positions of o and o' are independent of $o \prec_q^{[t_0,t]} o'$. Consequently, the probability $P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} | o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|}$, which corresponds to the probability of future positions of o and o' given current positions of o and o' is unaffected by conditioning to $o \prec_q^{[t_0,t]} o'$.

Using the law of total probability[2], we obtain

$$\sum_i P(o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|} | o \prec_q^{[t_0,t]} o') \cdot$$

$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} |$$
$$o(t) = s_{i \bmod |\mathcal{S}|} \wedge o(t) = s_{i \; div \; |\mathcal{S}|} \wedge o \prec_q^{[t_0,t]} o') =$$
$$P(o(t+1) = s_{j \bmod |\mathcal{S}|} \wedge o(t+1) = s_{j \; div \; |\mathcal{S}|} | o \prec_q^{[t_0,t]} o')$$

which equals the vector $(s_{o \times o'}(t+1))_j$ by its definition.

Given both a-priori probabilities and prior probabilities, we can now compute the reverse transition matrix $Ro \times o'(t)$ of o and o' using Equation 14.2. □

Lemma 40. *The computation of $P\forall NN(o, q, \mathcal{DB}, T)$ is NP-hard in $|\mathcal{DB}|$.*

Proof. By definition of predicate $(o \prec_q^T o_i)$, we can rewrite the probability that o is the nearest neighbor of q during T as follows:

$$P\forall NN(o, q, \mathcal{DB}, T) = P(\forall o_i \in \mathcal{DB} : o \prec_q o_i) = P(\bigwedge_{o_i \in \mathcal{DB}} o \prec_q o_i). \tag{14.4}$$

Clearly, Equation 14.4 follows from the fact that o is the NN of q if and only if o is closer to q than all other objects in $\mathcal{DB}$ during time T. Using the chain rule of probability, which iteratively uses the rule $P(A \wedge B) = P(A) \cdot P(B|A)$ for conditional probabilities, we obtain

$$P(\bigwedge_{o_i \in \mathcal{DB}} o \prec_q o_i) = \tag{14.5}$$
$$P(o \prec_q o_{|\mathcal{DB}|} | o \prec_q o_1 \wedge ... \wedge o \prec o_{|\mathcal{DB}|-1})$$
$$\cdot ... \cdot P(o \prec_q o_1)$$
$$= \prod_{1 \leq i \leq \mathcal{DB}} P(o \prec_q o_i | \bigwedge_{j<i} o \prec_q o_j).$$

To compute the conditional probabilities in the above equation, we can not exploit any independence. In particular, events $(o \prec_q o_1)$ and $(o \prec_q o_2)$ are mutually dependent, even though objects o, o_1 and o_2 are assumed to be (marginally) independent. This observation follows from the fact, that both random predicates $(o \prec_q o_1)$ and $(o \prec_q o_2)$ depend on the position of o. Semantically, the condition $(o \prec_q o_1)$ changes the distribution of the possible trajectories of o, by only allowing possible worlds where o is closer to q than o_1,

[2]The law of total probability states that for two random variables A and B, it holds that $P(A = a) = \sum_{b \in B} P(A = a | B = b) \cdot P(B = b)$. Introducing a fixed condition C yields $P(A = a|C) = \sum_{b \in B} P(A = a | B = b, C) \cdot P(B = b|C)$. In this case, the random variable A corresponds to the joint distribution of o and o' at time $t+1$, the random variable B corresponds to the joint distribution of o and o' at time t, and C corresponds to the event $o \prec_q^{[t_0,t]}$

thus "pressing" o closer to q, thus increasing the probability of o being closer to q than o_2. Thus, the random events $(o \prec_q o_i)$ and $(o \prec_q o_j)$ are positively correlated in general.

To show how this observation implies that P∀NN queries are NP-Hard, we investigate the complexity of the problem of conditioning the a-priori model $M_{k,i}^{prior}(t-1) = P(o(t) = s_i | o(t-1) = s_k)$ of an object o to the event that o dominates another object $o_a \in \mathcal{DB}$, yielding model $M_{k,i}^{post.}(t-1) = P(o(t) = s_i | o(t-1) = s_k, o \prec_q^T o_a)$. The problem here is, as we will see, that the adapted model derived using the forward-backward paradigm, albeit correct, does not satisfy the Markov property. Thus, the new model can not be used to for efficient inference, as the lack of Markov property results in exponential number of possible worlds to consider.

To compute $P(o(t) = s_i | o(t-1) = s_k, o \prec_q^T o_a), s_i, s_k \in \mathcal{S}$, the idea is to treat the positions of o and o_a as a single joint stochastic process, defined on the event space $\mathcal{S}^2$. Then, the joint a-priori transition matrix $M_{o \times o_a}(t)$ is conditioned to the event $o \prec_q^T o_a$ to obtain a joint probability matrix $M_{o \times o_a}(t-1) =$

$$P(o(t) = s_i, o_a(t) = s_j | o(t-1) = s_k, o_a(t-1) = s_l, o \prec_q^T o_a)$$

.

Next, we need to reduce this joint transition matrix to an adapted transition matrix $M_{k,i}^{post.}(t-1) = P(o(t) = s_i | o(t-1) = s_k, o \prec_q^T o_a)$ of object o. By applying the law of conditional probability, it can be shown that:

$$M_{k,i}^{post.}(t-1) = \sum_{s_j} \sum_{s_l} P(o_a(t-1) = s_l | o(t-1) = s_k, o \prec_q^T o_a) *$$

$$P(o(t) = s_i, o_a(t) = s_j | o(t-1) = s_k, o_a(t-1) = s_l, o \prec_q^T o_a)$$

Assuming that the Markov property still holds, we should get the same results for

$$P(o(t) = s_i | o(t-1) = s_k, \ldots, o(t-n) = s_{t-n}, o \prec_q^T o_a) =$$

$$\sum_{s_j} \sum_{s_l} P(o_a(t-1) = s_l | o(t-1) = s_k, \ldots, o(t-n) = s_{t-n}, o \prec_q^T o_a) *$$

$$P(o(t) = s_i, o_a(t) = s_j | o(t-1) = s_k, o_a(t-1) = s_l, o \prec_q^T o_a)$$

which is clearly not equivalent, i.e. the Markov property does not hold on the reduced transition matrices. □

14.3.3 The PCNN Query

The traditional CNN query [153, 182], retrieves the nearest neighbor of every point on a given query trajectory in a time interval T. This basic definition usually returns $m << |T|$ time intervals together having the same nearest neighbor. The main issue when considering uncertain trajectories and extending the query definition is the possibly large number of results due to highly overlapping and alternating result intervals. In particular, considering Definition 66, a PCNN result may have an exponential number of elements when τ becomes small. This is because in the worst case each $T_i \subseteq T$ can be associated with an object o for which the probability $P\forall NN(o, q, \mathcal{DB}, T_i) \geq \tau$, i.e., 2^T different T_i's occur in the result set.

To alleviate (but not solve) this issue, in the following we propose a technique based on Apriori pattern mining to return the subsets of T that have a probability greater than τ. This algorithm requires to compute a $P\forall NN$ probability in each validation step; we assume that this is achieved by employing the sampling approach proposed in Section 16. Since each subset of T may have a probability greater than τ (especially when τ is chosen too small), a worst-case of $O(2^n)$ validations may have to be performed.

Algorithm. Algorithm 10 shows how to compute, for a query trajectory q, a time interval T, a probability threshold τ, and an uncertain trajectory $o \in \mathcal{DB}$ all $T_i \subseteq T$ for which o is the nearest neighbor to q at all timestamps in T_i with probability of at least τ, and the corresponding probabilities.

Algorithm 10 $PC_\tau NN(q, o, \mathcal{DB}, \mathrm{T}, \tau)$

1: $L_1 = \{(\{t\}, P) | t \in T \wedge P = P\forall NN(o, q, \mathcal{DB} \setminus \{o\}, \{t\}) \geq \tau\}$
2: **for** $k = 2; L_{k-1} \neq \emptyset; k++$ **do**
3: $\quad X^k = \{T_k \subseteq T | |T_k| = k \wedge \forall T'_{k-1} \subset T_k \exists (T'_{k-1}, P) \in L_{k-1}\}$
4: $\quad L_k = \{(T_k, P) | T_k \in X^k \wedge P = P\forall NN(o, q, \mathcal{DB} \setminus \{o\}, T_k) \geq \tau\}$
5: **end for**
6: **return** $\bigcup_k L_k$

We take advantage of the anti-monotonicity property that for a T_i to qualify as a result of the PCNNQ query, all proper subsets of T_i must also satisfy this query. In other words if o is the P$\forall$NN of q in T_i with probability at least τ, then for all $T_j \subset T_i$ o must be the P$\forall$NN of q in T_j with probability at least τ. Exploiting this property, we adapt the Apriori pattern-mining approach from [8] to solve the problem as follows. We start by computing the probabilities of all single points of time to be query results (line 1). Then, we iteratively consider the set X^k of all timestamp sets with k points of time by extending timestamp sets T_{k-1} with an additional point of time $t \in T \setminus T_{k-1}$, such that all $T'_{k-1} \subset T_k$ have qualified at the previous iteration, i.e., we have $P\forall NN(o, q, \mathcal{DB} \setminus \{o\}, T'_{k-1})) \geq \tau$ (line 3).

The P$\forall$NN probability is monotonically decreasing with the number of points in time considered, i.e., $\mathrm{P}\forall\mathrm{NN}(o, q, \mathcal{DB} \setminus \{o\}, T_k) \geq \mathrm{P}\forall\mathrm{NN}(q, \mathcal{DB}, T_{k+1})$ where $T_k \subset T_{k+1}$. Therefore we do not have to further consider the set of points of time T_k that do not qualify for the

next iterations during the iterative construction of sets of time points. Based on the sets of timesteps T_k constructed in each iteration we compute $P\forall NN(o, q, \mathcal{DB} \setminus \{o\}, T_k)$ to build the set of results of length k (line 4) that are finally collected and reported as result in line 6. The basic algorithm can be sped up by employing the property that given $\mathrm{P}\forall\mathrm{NN}(o, q, \mathcal{DB} \setminus \{o\}, T_1) = 1$ the probability of $\mathrm{P}\forall\mathrm{NN}(o, q, \mathcal{DB} \setminus \{o\}, T_1 \cup T_2) = \mathrm{P}\forall\mathrm{NN}(o, q, \mathcal{DB} \setminus \{o\}, T_2)$.

Based on Algorithm 10 it is possible to define a straightforward algorithm for processing PCNNQ queries (by considering each object o' from the database).

Due to the high complexity of all three defined semantics of kNN queries, an approximate solution is presented in Chapter 16

Chapter 15

Indexing Uncertain Spatio-Temporal Data

In this chapter, an index structure that facilitates efficient processing of spatio-temporal window queries based on an appropriate model for uncertain trajectories is proposed. The proposed hierarchical index uses novel approximation techniques in order to probabilistically bound the uncertain movement of objects; these techniques allow for efficient and effective filtering during query evaluation. To the best of our knowledge, this is the first approach that supports query evaluation on very large uncertain spatio-temporal databases, adhering to possible worlds semantics. We experimentally show that it accelerates the existing, scan-based approach by orders of magnitude. The next section, Section 15.1 presents (conservative) spatio-temporal as well as probabilistic uncertain spatio-temporal (UST-) object approximations, which serve as building blocks for our proposed index, to be presented in Section 15.2.

15.1 Approximating Uncertain Spatio-Temporal Objects

In order to design an index for uncertain spatio-temporal objects to speed-up probabilistic spatio-temporal queries we first have to identify appropriate search keys that can be used to determine result candidates quickly. For this purpose we define approximations of the spatio-temporal space in which an object $o \in \mathcal{DB}$ can be located in. Thereby, we will concentrate on approximating the uncertain trajectory of an object between two observations $(o(t_i), o(t_j), t_i, t_j \in o.T_{obs})$. In the following, we first introduce (conservative) spatio-temporal as well as probabilistic uncertain spatio-temporal (UST-) object approximations and show how these approximations can be exploited to identify efficiently result candidates and *true drops*.

15.1.1 UST-Object Approximation

To bound the possible locations of an object o between two subsequent observations $(o(t_i), o(t_j))$, we need to determine all state-time pairs $(s,t) \in S \times T, t_i \leq t \leq t_j$ such that o has a non-zero probability of being at state s at time t. This is done by considering all possible paths between state $o(t_i)$ at time t_i and state $o(t_j)$ at time t_j. An example of a small set of such paths is depicted in Figure 15.1(a). Here, we can see a set of five possible trajectories of an object o, i.e. all possible $(state, time)$ pairs of o in the time interval $[t_i, t_j]$. In practice, the number of possible paths becomes very large. Nonetheless, we can efficiently compute the set of possible $(state, time)$ pairs using the Markov-chain model: The set of state-time pairs S_i reachable from $o(t_i)$ can be computed by performing $t_j - t_i$ transitions using the Markov chain $o.M(t)$ of o, starting from state $o(t_i)$ and memorizing all reachable $(state, time)$ pairs. Similarly, we can compute S_j as all state-time pairs $(s,t) \in S \times \mathcal{T}, t_i \leq t \leq t_j$ such that o can reach state $o(t_j)$ at time t_j by starting from state s at time t. S_j can be computed in a similar fashion, starting from state $o(t_j)$ and using the transposed Markov chain $o.M(t)^T$. The intersection $S_{i,j} = S_i \cap S_j$ yields all state-time pairs which are consistent with both observations. Let us note that in practice, it is more efficient to compute S_i and S_j in a parallel fashion, to reduce the explored space. When the computation of S_i and S_j meet at some time $t_i \leq t \leq t_j$, we can prune any states which are not reachable by both $s(t_i)$ at time t_i and $s(t_j)$ at time t_j. However, the number $|S_{i,j}|$ of possible state-time pairs in $S_{i,j}$ can grow very large, so it is impractical for our index structure (proposed in Section 15.2) to store all $S_{i,j}$ for each $o \in DB$ in our index structure. Thus, we propose to conservatively approximate $S_{i,j}$. The issue is to determine an appropriate approximation of $S_{i,j}$ which tightly covers $S_{i,j}$, while keeping the representation as simple as possible. The basic idea is to build the approximation by means of both object observations $o(t_i)$ and $o(t_j)$ with the corresponding velocity of propagation in each dimension. To do so, we first compute for the set of state-time pairs S_i to derive the maximum and minimum possible velocity in the time interval $[t_i, t_j]$:

$$v_d^{\leqslant} := max_{(s,t) \in S_i}\left(\frac{s[d] - o(t_i)[d])}{t - t_i}\right)$$

$$v_d^{\leqslant} := min_{(s,t) \in S_i}\left(\frac{s[d] - o(t_i)[d])}{t - t_i}\right)$$

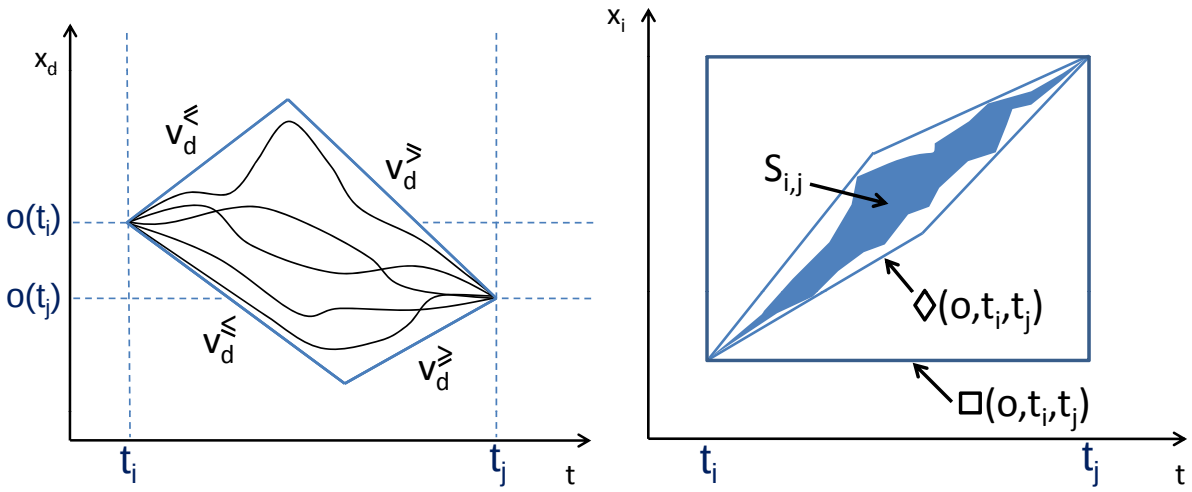
where $s[d]$ ($o(t_i)[d]$) denotes the projection of state s ($o(t_i)$) to the d-th dimension. By definition, we can guarantee that for any $t_i \leq t \leq t_j$ it holds that

$$o(t)[d] \leq o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant} \text{ and}$$

$$o(t)[d] \geq o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}$$

Furthermore, we bound the velocity of propagation at which o can have reached state $o(s_j)$ at time t_j from each location in the state-space S_j:

$$v_d^{\geqslant} := max_{(s,t) \in S_j}\left(\frac{o(t_j)[d] - s[d]}{t_j - t}\right)$$



(a) Approximating trajectories (b) Diamond vs MBR

Figure 15.1: Spatio-Temporal Approximation.

$$v_d^{\geqslant} := min_{(s,t)\in S_j}(\frac{o(t_j)[d] - s[d]}{t_j - t})$$

again, we can bound the position of o in dimension $1 \leq d \leq D$ at time $t_i \leq t \leq t_j$ as follows:

$$o(t)[d] \leq o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant}, \text{ and}$$
$$o(t)[d] \geq o(t_j)[d] - (t_j - t) \cdot v_d^{\leqslant}$$

In summary, using the positions $o(t_i)$ at time t_i and $o(t_j)$ at time t_j, and using velocities $v_d^{\leqslant}, v_d^{<}, v_d^{\geqslant}, v_d^{>}$, we can bound the random variable of the position $o(t)$ of o at time $t_i \leq t \leq t_j$ by the interval

$$o(t)[d] \in I_d(t) := [max(o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}, o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant})),$$

$$min(o(t_i)[d] + (t - t_i) \cdot v_d^{\leqslant}, o(t_j)[d] - (t_j - t) \cdot v_d^{\geqslant})] \tag{15.1}$$

Deriving these intervals for each dimension, yields an axis-parallel rectangle, approximating all possible positions of o at time t. In the following, we will call this time dependent spatial approximation of $o(t)$ in the time interval $[t_i, t_j]$ between two observations $o(t_i)$ and $o(t_j)$ a spatio-temporal *diamond*, denoted as $\diamond(o, t_i, t_j)$.[1] A nice geometric property of this approximation is that computing the intersection with the query window at each time t is very fast. Another advantage is that existing spatial access methods (e.g., R-trees) can be easily used to efficiently organize these approximations. To store the approximation, we

[1]Our definition of a diamond differs from related work, since the 4 sides of the polygon (when projected to each dimension) may differ in length, so the polygon may not be a rhombus, but in practice, the polygon does resemble a diamond in most cases.

only need to store the $4 \cdot D$ real values $v_d^{\lessdot}, v_d^{\leqslant}, v_d^{\gtrdot}, v_d^{\geqslant}, 1 \leq d \leq D$.[2] A diamond is reminiscent to a time-parameterized rectangle, used to model the worst-case MBR for a set of moving objects in [160]; however, the way of deriving velocities is different in our case. As an example, Figure 15.1(a) shows for one dimension $d \in D$, positions $o(t_i)$ at time t_i and $o(t_j)$ at time t_j. The diamond formed by the velocity bounds $v_d^{\lessdot}, v_d^{\leqslant}, v_d^{\geqslant}$ and $v_d^{\gtrdot}$ conservatively approximates the possible $(location, time)$ pairs.

Note that it is possible to use a minimal bounding rectangle $\square(o, t_i, t_j)$ instead of the diamond $\diamond(o, t_i, t_j)$ to conservatively approximate the $(location, time)$ space $S_{i,j}$. In cases, however, where the movement of an object in one dimension is biased in one direction, a rectangle may yield a very bad approximation (see Figure 15.1(b) for an example). Our index employs both approximations $\square(o, t_i, t_j)$ and $\diamond(o, t_i, t_j)$ for spatio-temporal pruning; $\square(o, t_i, t_j)$ is used for high-level indexing and filtering, while $\diamond(o, t_i, t_j)$ is used as a second-level filter.

15.1.2 Spatio-Temporal Filter

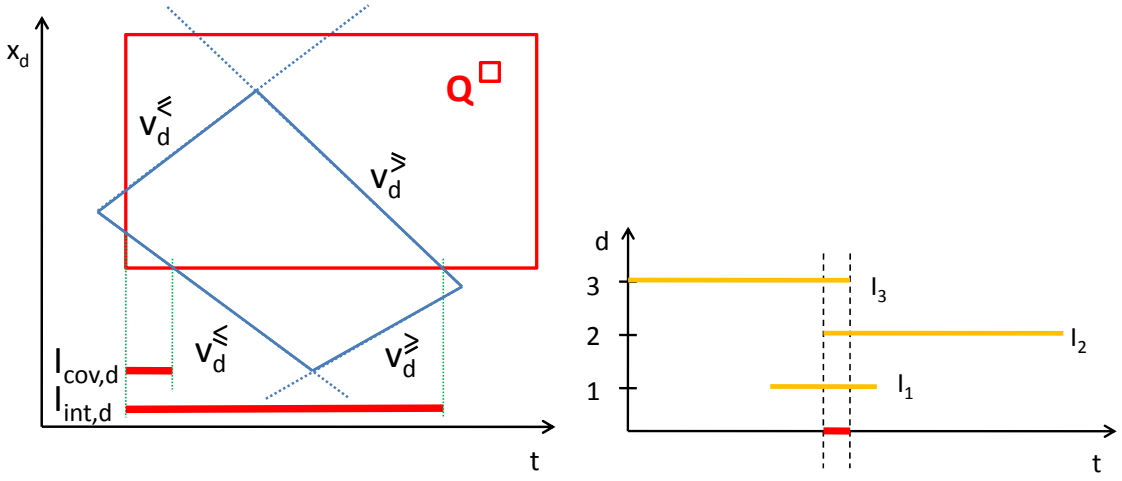
Based on the spatio-temporal approximation of an uncertain object as described in the previous section, it is possible to perform filtering during query processing. In the case of a $PST\tau\exists Q$ query, if each diamond assigned to an object $o \in \mathcal{DB}$ does not intersect the query window, then o is safely pruned. In turn, if a diamond of o is inside the query window $S^{\square}$ in space, i.e. fully covered by $S^{\square}$, at any point of time $t \in T^{\square}$, then o is a *true hit* and, thus, o can be immediately reported as result of the query. Similarly, for a $PST\tau\forall Q$ query, objects whose diamonds do not intersect the query window during the *entire* query time-range $T^{\square}$ can be pruned. On the other hand, an object o with a diamond is fully covered by $S^{\square}$ for each point of time $t \in T^{\square}$. is immediately reported as a *true hit*.

In order to employ the above spatio-temporal pruning conditions, for a diamond $\diamond(o, t_i, t_j)$ of an object o we need to determine the points of time when it intersect the query window $S^{\square}$ in space, as well as the points of time when $\diamond(o, t_i, t_j)$ is fully covered by $S^{\square}$. For this purpose, it is helpful to focus on the spatial domain $\mathcal{S}$ and interpret a diamond as well as the query as a time-parameterized (moving) rectangle. By doing so, we can adapt the techniques proposed in [160]: In general, a rectangle R_1 intersects (covers) another rectangle R_2, if and only if R_1 intersects (covers) R_2 in each dimension. Thus, for each spatial dimension d ($d \in \{1, \ldots D\}$), we compute the points of time when the extents of the rectangles intersect in that dimension and the points of time when the extents of the diamond rectangle is fully covered by the query rectangle $S^{\square}$.

Regarding a single dimension d, with Equation 15.1 the query window given by $Q_d^{\square}$ intersects the diamond given by $o(t_i)[d]$, $o(t_j)[d]$, $v_d^{\lessdot}$, $v_d^{\leqslant}$, $v_d^{\gtrdot}$ and $v_d^{\geqslant}$ within the points of time

$$I_{int,d} := \{t \in (T^{\square} \cap [t_i, t_j] | I_d(t) \cap S_d^{\square}\}.$$

[2] The observations $(o(t_i), t_j)$ and $o(t_j), t_j)$ which are furthermore required to span a diamond, are already stored in $\mathcal{DB}$.



(a) query intersection (b) dimension-wise intersections

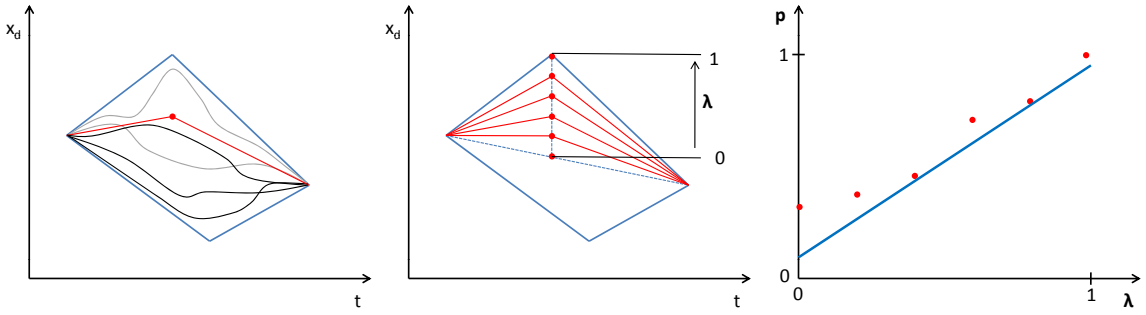
Figure 15.2: Intersection between query and diamond

Similar, $Q_d^{\square}$ fully covers the diamond within the points of time

$$I_{cov,d} := \{t \in T^{\square} \cap [t_i, t_j] | I_d(t) \subseteq S_d^{\square}\}.$$

An example is illustrated in Figure 15.2(a). To compute both sets $I_{int,d}$ and $I_{cov,d}$, we intersect the margins of the diamond with the query window resulting in a set of time intervals, which subsequently have to be intersected accordingly in order to derive $I_{int,d}$ and $I_{cov,d}$. Now, let us consider the overall intersection time interval $I_{int} = \bigcap_{d=1}^{D} I_{int,d}$ (e.g., see Figure 15.2(b)) and the overall points of covering time $I_{cov} = \bigcap_{d=1}^{D} I_{cov,d}$. In the case of a $PST\tau\exists Q$, if for an object $o \in \mathcal{DB}$ there is no diamond yielding a non-empty set I_{int}, o can be safely pruned. If any diamond of o yields a non-empty set I_{cov}, o can be reported as result. In the case of a $PST\tau\forall Q$, an object $o \in \mathcal{DB}$ can be safely rejected if all I_{int} of all diamonds of o together do not completely cover the query time range $T^{\square}$. Contrary, if all I_{cov} of all diamonds of o together completely cover $T^{\square}$, o can be reported as a result of $PST\tau\forall Q$.

In summary, the spatio-temporal filter can be used to identify uncertain object trajectories having a probability of 100% or 0% intersecting (remaining in) the query region $Q^{\square}$. Still, the probability threshold τ of the query is not considered by this filter. In addition, the object approximation may cover a lot of dead space if there exist outlier state-time pairs which determine one or more of the velocities, despite having a very low probability. In the following, we show how to exclude such unlikely outliers in order to shrink the approximation, while maintaining probabilistic guarantees that employ the probability threshold τ.



(a) A sub-diamond (b) Possible sub-diamonds (c) linear optimization

Figure 15.3: Construction of Probabilistic Diamonds

15.1.3 Probabilistic UST-Object Approximation

In the next filter step, we will make a first approach at bounding the actual probability that a uncertain spatio-temporal object o intersects the query window. Therefore, we make the following observation: In many applications, the set of possible paths within a diamond is generally not uniformly distributed. In many applications, paths that are close to the direct connection between the observed locations are more likely than extreme paths along the edges of the diamond. For example, when a person as been observed at position *kitchen* at two times t_i and t_j ($t_j > t_i$), it is much more likely, that she simply remained at *kitchen* in the time between t_i and t_j, than the extreme alternative of having sprinted out of the kitchen as fast as possible, then having turned around to sprint back in order to reach the kitchen just in time t_j. On a road network, a vehicle having a maximum speed of $50mph$, is much more like to having travelled at a constant speed of $40mph$, rather than first having driven backwards at $-10mph$, then switching to full pace forward to barely reach t_j by driving $50mph$.

We now propose a tighter approximation, based on the intuition that the set of possible paths within a diamond is generally not uniformly distributed: paths that are close to the direct connection between the observed locations often are more likely than extreme paths along the edges of the diamond. Therefore, given a query with threshold τ, we could take advantage of a tighter approximation, which bounds all paths with cumulative probabilities τ to perform more effective pruning.

Based on this idea, we exploit the Markov-chain model in order to compute new diamonds, which are spatio-temporal subregions, called subdiamonds, of the (full) diamond $\diamond(o, t_i, t_j)$, as depicted in Figure 15.3(a). For each such subdiamond, we will then show how to compute the cumulative probability of all possible trajectories of o passing only through this subdiamond. Let us focus on restricting the diamond at one direction of one dimension; we choose one dimension $d \in D$, and one direction $dir \in \{\wedge, \vee\}$. Direction $\wedge$ ($\vee$) corresponds to the two diamond sides $v_d^{\leqslant}$ and $v_d^{\geqslant}$ ($v_d^{\leqslant}$ and $v_d^{\geqslant}$). To obtain the subdiamond, we scale the corresponding sides by a factor $\lambda \in [0, 1]$ relative to the average velocity

$v_d^{avg} = \frac{o(t_j)[d]-o(t_i)[d]}{t_j-t_i}$. We obtain the adjusted velocity values for direction $\wedge$ as follows:

$$v^{\leqslant\lambda}_d = ((v_d^{\leqslant} - v_d^{avg}) \cdot \lambda) + v_d^{avg}$$
$$= v_d^{\leqslant} \cdot \lambda + v_d^{avg} \cdot (1 - \lambda)$$

and

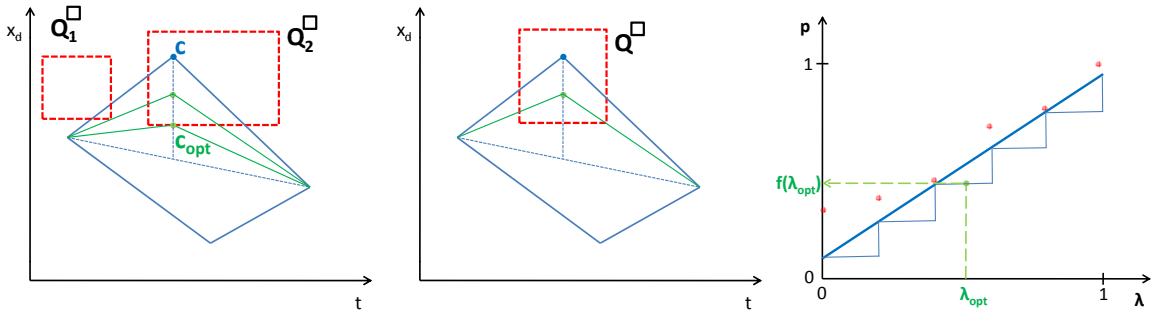
$$v^{\geqslant\lambda}_d = ((v_d^{\geqslant} - v_d^{avg}) \cdot \lambda) + v_d^{avg}$$
$$= v_d^{\geqslant} \cdot \lambda + v_d^{avg} \cdot (1 - \lambda)$$

The adjusted velocity values for direction $\vee$ can be computed analogously. Thus, for a given diamond $\diamond(o, t_i, t_j)$, dimension $d \in D$, direction $dir \in \{\wedge, \vee\}$ and scalar $\lambda \in [0, 1]$, we obtain a smaller diamond $\diamond(o, t_i, t_j, d, dir, \lambda)$, derived from $\diamond(o, t_i, t_j)$ by scaling direction dir in dimension d by a factor of λ. Figure 15.3(b) illustrates some subdiamonds for one dimension, the $\wedge$ direction and for various values of λ.

To use such subdiamonds for probabilistic pruning, we first need to compute the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ that object o will remain within $\diamond(o, t_i, t_j, d, dir, \lambda)$ for the whole time interval $[t_i, t_j]$, in a correct and efficient way. The main challenge for correctness, is to cope with temporal dependencies, i.e. the fact that the random variables $o(t_i)$ and $o(t_i + \delta t)$ are highly correlated. Thus, we cannot simply treat all random variables $o(t)$ as mutually independent and aggregate their individual distributions. To illustrate this issue, consider Figure 15.3(a), where one sub diamond is depicted. Assume that each of the five possible trajectories has a probability of 0.2. We can see that three trajectories are completely contained in the subdiamond, so that the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ that o fully remains in the subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ is 60%. However, multiplying for each time instants $t \in [t_i, t_j]$ the individual probabilities that o is located in $\diamond(o, t_i, t_j, d, dir, \lambda)$ at time t, may produce an arbitrary small result, and generally wrong result, depending on the number of time instants in $[t_i, t_j]$.

Also, an approach that simply computes the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ by multiplying the probabilities at each point of time $t \in [t_i, t_j]$ is incorrect. This is clear, since this probability becomes arbitrarily small as the number of time steps between t_i and t_j increases. This wrong result can be accounted to the interdependencies between the random variables $o(t)$, which are ignored by multiplication of their respective probabilities to be inside the subdiamond.

Furthermore, due to the generally exponential number of possible trajectories, the probability $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ is too expensive to compute by iterating over all possible trajectories. Instead, we propose the following approach to compute this probability efficiently and correctly.



(a) Computing λ_{opt} ($\exists$-case) (b) Computing λ_{opt} ($\forall$-case) (c) conservative approximation

Figure 15.4: Linear Probabilistic Diamond Approximation

Computing the Probability of a Subdiamond

To identify the probability of possible trajectories between $o(t_i)$ at t_i and $o(t_j)$ at t_j that are completely contained in $\diamond(o, t_i, t_j, d, dir, \lambda)$, an intuitive approach is to start at $o(t_i)$ at time t_i and perform $t_j - t_i$ transitions using the Markov-chain $o.M(t)$. After each transition, we identify states that are outside $\diamond(o, t_i, t_j, d, dir, \lambda)$. Any possible trajectory which reaches such a state is flagged. Upon reaching t_j, we only need to consider possible trajectories in state $o(t_j)$, since all other worlds have become impossible due to the observation of o at t_j. The fraction of un-flagged worlds at state $o(t_j)$ at time t_j yields the probability that o does not completely remain in $\diamond(o, t_i, t_j, d, dir, \lambda)$. Since the context is clear, the rest of this section, we simply use $\diamond$ to denote $\diamond(o, t_i, t_j, d, dir, \lambda)$.

To formalize the above approach, we first rewrite the probability $P(inside(o, \diamond)|o(t_i), o(t_j)))$ that the trajectory of o remains in the subdiamond $\diamond$, by explicitly notion of the given the observations $o(t_i)$, $o(t_j)$ at times $t_i, t_j \in o.T_{obs}$ applying the definition of conditional probability:

$$P(inside(o, \diamond)|o(t_i), o(t_j)) = \frac{P(o(t_j)|inside(o, \diamond), o(t_i))}{P(o_{t_j}|o_{t_i})},$$

where $P(o(t_j)|inside(o, \diamond), o(t_i))$ denotes the probability that o reaches the state $o(t_j)$ observed by observation $o(t_j)$, given that o, starting at $o(t_i)$ at time t_i remains inside $\diamond$. $P(o(t_j)|o(t_i))$ denotes the probability that state $o(t_j)$ at time t_j is reached, given that o starts at $o(t_i)$ at time t_i, regardless whether o remains in $\diamond$. To compute the latter two probabilities, we proceed as follows. Instead of a single probability vector p_t of length $|S|$, describing the state distribution of o at time t, we use two probability vectors p_t^+ and p_t^-, each of length $|S|$. An entry $p_t^+(i)$ corresponds to the joint probability of the event that o is located at state i at time t and the event $inside(o, \diamond, t)$ that o has (so far) been completely contained in the diamond in the time interval $[t_i, t]$. Analogously, an entry $p_t^-(i)$ corresponds to the probability that o is located at state i at time t and has already left the diamond at, or before, time t. Thus, vectors p_t^+ and p_t^- describe the probability density function of o on the two dimensional event space $(state \times inside(o, \diamond, t))$, where

the random variable $inside(o, \diamond, t)$ is dichotomous (i.e. true or false). Then, we proceed as follows: Starting at t_i, we perform $t_j - t_i$ transitions using alternated Markov-chains, using both vectors p_t^+ and p_t^-. Initially, $p_{t_i}^+$ is a unit vector, having $p_{t_i}^+(o(t_i)) = 1$ (and all other entries set to zero) and $p_{t_i}^-$ is the zero vector. Semantically, this means that at time t_i, o must be in state $o(t_i)$ with a probability of one, and must so far have retained to $\diamond$. At each transition from t_k to t_{k+1}, we decide for each reachable state $s \in S$, whether s is located in $\diamond$ at time t_{k+1}. This decision can be easily made by simply testing whether the coordinate $s[d]$ of state s in dimension d falls into interval of $\diamond$ using Equation 15.1. In the following, let $in(s, t, \diamond)$ be an indicator function that returns 1 if the state/time pair (s, t) is inside $\diamond$, and 0 otherwise. For each state s where $in(s, t_{k+1}, \diamond) = 0$, we add the probability of $p_t^+(s)$ to $p_t^-(s)$ and set $p_t^+(s)$ to zero. That is, in each iteration we compute

$$p_{t_{k+1}}^+ = p_{t_k} \cdot M, p_{t_{k+1}}^- = p_{t_k} \cdot M$$

and then for each state $1 \leq s \leq |S|$

$$p_{t_{k+1}}^-(s) = p_{t_{k+1}}^-(s) + p_{t_{k+1}}^+(s) \cdot (1 - in(s, t, \diamond)),$$

$$p_{t_{k+1}}^+(s) = p_{t_{k+1}}^+(s) \cdot in(s, t, \diamond)$$

At the final time t_j, value $p_{t_j}^+$ corresponds to $P(o(t_j)|inside(o, \diamond))$; i.e., the probability of o having reached the observed state $o(t_j)$ at time t_j, without having left $\diamond$, and the sum of $p_{t_j}^+$ and $p_{t_j}^-$ corresponds to the probability $P(o_{t_j})$ that $o(t_j)$ is reached at all. Thus:

$$P(inside(o, \diamond)) = \frac{p_{t_j}^+(o(t_j))}{p_{t_j}^+(o(t_j)) + p_{t_j}^-(o(t_j))}$$

The overall time for computing the probability of a subdiamond $\diamond$ is in $O((t_j - t_i) \cdot |S|)$, since in each of the $t_j - t_i$ iterations, we only need to consider a finite number of $2|S|$ states. Since M is a sparse matrix, the vectors p_t^+ and p_t^- remain sparse as well. This observation allows to further accelerate the computation of $P(inside(o, \diamond))$ using sparse matrix operations. Also note, that sampling methods are of limited use for computing $P(inside(o, \diamond)$. In addition to the approximative nature of sampling, most sampled paths derived by starting at $o(t_i)$ at time t_i using the Markov chain M, will not reach state $o(t_j)$ at time t_j. Thus, a very large sample of paths has to be sampled, in order to find a representative number of paths that intersect $o(t_j)$ at time t_j.

15.1.4 Finding the optimal Probabilistic Diamond

In the previous section, we described, how to compute the probability of a probabilistic diamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ from a diamond $\diamond(o, t_i, t_j)$, dimension d, direction dir, and scaling factor λ. In this section we will show how to find, for a given query window $Q^\Box$ and a given query predicate the sub-diamond with the highest pruning power. Let us focus on PST$\tau\exists$ queries first. That is, our aim is to find a value for d, dir and λ, such that the resulting subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda)$ does not intersect $Q^\Box$, and at the same time it has a high probability $P(inside(o, \diamond))$. This probability can be used to prune o as we will show later. Formally, we want to efficiently determine

$$argmax_{d\in D, dir\in\{\vee,\wedge\},\lambda\in[0,1]}[P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))]$$

constrained to $Q^\Box \cap \diamond(o, t_i, t_j, d, dir, \lambda) = \emptyset$.

For a single dimension d, and the *north* direction, a possible situation is depicted in Figure 15.4(a). Here, the projection $\diamond_d(o, t_i, t_j)$ of the full diamond $\diamond(o, t_i, t_j)$ to the d'th dimension and the projections $Q_1^\Box[d]$ and $Q_2^\Box[d]$ of two query windows $Q_1^\Box$ and $Q_2^\Box$ are depicted. The aim is to find the largest values λ_{opt} of λ, such that the corresponding probabilistic diamond $\diamond(o, t_i, t_j, d, \wedge, \lambda_{opt}^\exists)$ which we call optimal subdiamond, does not intersect $Q_1^\Box$ ($Q_2^\Box$). To solve this problem, we distinguish between the following cases.

Case 1: the direct line between observations $(o(t_i), t_i)$ and $(o(t_j), t_j)$ in dimension d intersects $Q^\Box[d]$. In this case, there cannot exist any $\lambda \in [0, 1]$ such that $Q^\Box \cap \diamond(o, t_i, t_j, d, dir, \lambda) = \emptyset$. Therefore, our problem has no solution in dimension d, which is ignored. In this case there exists no probabilistic diamond for dimension d in direction $\wedge$. If this is the case for each dimension, i.e. if the direct line between $(o(t_i), t_i)$ and $(o(t_j), t_j)$ intersects $Q^\Box$ in the full space, then we cannot find any useful probabilistic subdiamond. Intuitively, for such a diamond the true probability of intersecting $Q^\Box$ (after refinement), is expected to be very large, thus $\diamond(o, t_i, t_j)$ is unlikely to subjectable to probabilistic pruning, and thus we skip it entirely in the probabilistic pruning step of our algorithm.

Case 2: the direct line between $(o(t_i), t_i)$ and $(o(t_j), t_j)$ does not intersect $Q^\Box[d]$, and we assume without loss of generality that $Q^\Box[d]$ is located above this line.[3] In addition, in this case, the time value of the north corner c of $\diamond_d(o, t_i, t_j)$ is located in the interval $T^\Box$ (e.g., see $Q_2^\Box$ in Figure 15.4(a)).[4] In this case, the edge $v_{opt}^{\leqslant}$ of the optimal subdiamond $\diamond(o, t_i, t_j, d, dim, \lambda_{opt}^\exists)$ is given by $(o(t_i), t_i)$ and (s, t) where s correspond to the lower bound of $S^\Box[d]$ and t is equal the time component of c.

Case 3: $Q^\Box$ is above the direct line between $(o(t_i), t_i)$ and $(o(t_j), t_j)$ (as in Case 2), but the time value of the north corner c of $\diamond_d(o, t_i, t_j)$ is not located in the time interval $T^\Box$ (e.g. $Q_1^\Box$ of Figure 15.4(a)). In this case, the optimal subdiamond must touch a corner of $Q^\Box[d]$ due to convexity of both $Q^\Box[d]$ and any diamond. If $Q^\Box[d]$ is located to the left of c (the right direction is handled symmetrically), then the edge $v_{opt}^{\leqslant}$ of the optimal subdiamond is given by the line between $(o(t_i), t_i)$ and the lower right corner of $Q^\Box[d]$ (c.f. Fig. 15.4(a)).

[3] If $Q^\Box[d]$ is below the line, we consider direction $dir = \vee$ symmetrically.

[4] Corner c is given by the intersection of lines $(o(t_i), t_i) + v^{\leqslant}$ and $(o(t_j), t_j) + v^{\geqslant}$.

The optimal value $\lambda^{\exists}_{opt}$ for cases 2 and 3 equals the quotient $\frac{v^{\leqslant}_{opt} - v_{avg}}{v^{\leqslant} - v_{avg}}$, i.e., the fraction of the maximum velocity of the optimal subdiamond and the maximum velocity of the full diamond, both normalized by the average velocity $v_{avg} = \frac{s(t_j)[d] - s(t_i)[d]}{t_j - t_i}$. After identifying the value for $\lambda^{\exists}_{opt}$, for a dimension d and a direction dir, we can compute the probability of the corresponding subdiamond $\diamond(o, t_i, t_j, d, dir, \lambda^{\exists}_{opt})$. Since we can guarantee, that any path in this subdiamond does not intersect the query window, we can obtain an lower bound probability

$$P_{LB}(never(o, t_i, t_j, Q^{\square})) = P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda^{\exists}_{opt}))) \tag{15.2}$$

of the event that o never intersects the query window in the time interval $[t_i, t_j]$. This directly yields an upper bound probability

$$P_{UB}(sometimes(o, t_i, t_j, Q^{\square})) =$$
$$1 - P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda^{\exists}_{opt}))) \tag{15.3}$$

of the reverse event that o intersects the query window at least once in $[t_i, t_j]$. This bound can be used for probabilistic pruning for PST$\tau\exists$ queries, as we will see in Section 15.1.6.

For PST$\tau\forall$ queries we can naively use the probabilistic subdiamond computed above, exploiting the fact that any world that does not qualify a PST$\tau\exists$ query $Q^{\square}$, cannot satisfy the corresponding PST$\tau\forall$ query $Q^{\square}$. Still, we can do better: We can extend the probabilistic subdiamond (i.e., increase the value of λ), until it becomes possible that a path in the subdiamond remains completely in $Q^{\square}$. In Case 1, where v_{exp} intersects $Q^{\square}$, again we can find no probabilistic subdiamond; in both other cases, we find the best probabilistic diamond for each dimension and each direction, as the maximum $\lambda^{\forall}_{opt}$, such that the resulting subdiamond does not contain both corners of $Q^{\square}[d]$ facing $\diamond(o, t_i, t_j)$. An example is given in Figure 15.4(b). Here, $\lambda^{\forall}_{opt}$ is chosen for the north direction, such that $v^{\geqslant}$ meets the lower right corner of $Q^{\square}[d]$. The resulting diamond $\diamond(o, t_i, t_j, d, dir, \lambda^{\forall}_{opt})$ is guaranteed to not contain any path that satisfies a PST$\tau\forall$ query. The reason is that $\lambda^{\forall}_{opt}$ is chosen such that there is guaranteed to be one point of time [5] when the query window $Q^{\square}$ is not intersected by $\diamond(o, t_i, t_j, d, dir, \lambda^{\forall}_{opt})$.

This observation allows to derive the lower bound probability

$$P_{LB}(sometimes\_not(o, t_i, t_j, Q^{\square}))$$
$$= P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda^{\forall}_{opt})))$$

of the event that o misses the query window at least once. Again, we derive an upper bound probability

$$P_{UB}(always(o, t_i, t_j, Q^{\square}))$$
$$= P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda^{\forall}_{opt})))$$

of the reverse event that o is always located in $Q^{\square}$, which can be used for pruning for PST$\tau\exists$ queries (see Section 15.1.6).

[5] corresponding to the corner which is not intersected by $\diamond(o, t_i, t_j, d, dir, \lambda^{\forall}_{opt})$

15.1.5 Approximating Probabilistic Diamonds

The main goal of our index structure, proposed in Section 15.2, is to avoid doing expensive probability computations for subdiamonds. Since the query window is not known in advance, $2D$ computations (i.e., one for each dimension and direction) have to performed in order to identify the optimal subdiamond for a given query and candidate object o. To avoid these computations at run-time, we propose to pre-compute, for each diamond $\diamond(o, t_i, t_j)$ in $\mathcal{DB}$, probabilistic subdiamonds for each dimension and direction and for a set Λ of λ-values. This yields a catalogue of probability values, i.e. a probability for each $\diamond(o, t_i, t_j, d, dir, \lambda), d \in D, dir \in \{\vee, \wedge\}, \lambda \in \Lambda$.

Given a query window, the optimal value λ_{opt} computed in Section 15.1.4 may not be in Λ. Thus, we need to conservatively approximate the probability of probabilistic diamonds $\diamond(o, t_i, t_j, d, dir, \lambda)$ for which $\lambda \notin \Lambda$. We propose to use a conservative linear approximation of $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda)))$ which increases monotonically with λ, using the precomputed probability values. For example, Figure 15.3(c), shows the $(\lambda, probability)$-space, for six values $\Lambda = \{0, 0.2, 0.4, 0.6, 0.8, 1\}$. The corresponding precomputed pairs $(\lambda, P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))))$ are depicted. Our goal is to find a function $f(\lambda)$ that minimizes the error with respect to $P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))$, while ensuring that $\forall \lambda \in [0, 1] : f(\lambda) \leq P(inside(o, \diamond(o, t_i, t_j, d, dir, \lambda))$. The latter constraint is required to maintain the conservativeness property of the approximation, which will be required for pruning. We model this as a linear programming problem: find a linear function $l(\lambda) = a \cdot \lambda + b$ that minimizes the aggregate error with respect to the sample points, under the constraint that the approximation line does not exceed any of the sample values (e.g., the line in Figure 15.3(c)). That is, we compute:

$$argmin_{a,b}(\sum_{\lambda \in \Lambda} P(\lambda) - (a + b \cdot \lambda))$$

subject to:

$$\forall \lambda \in \Lambda : P(\lambda) \geq a + b \cdot \lambda$$

We use the simplex algorithm to solve fast this optimization problem.

In summary, a probabilistic spatio-temporal object o is approximated by a set of $|o.T_{obs}| - 1$ diamonds, one for each subsequent time points $t_i, t_j \in o.T_{obs}$. Each diamond approximation contains the following:

- The spatio-temporal diamond $\diamond(o, t_i, t_j)$, consisting of four real values $v^{\leqslant}, v^{\leqslant}, v^{\geqslant}, v^{\geqslant}$.
- A set of $2 \cdot D$ linear approximation functions $f_{d,dir}(\lambda)$, one for each dimension $d \in D$ and each direction $dir \in \{\vee, \wedge\}$.

Next, we will show how to use these object approximations for efficient query processing over uncertain spatio-temporal data.

15.1.6 Probabilistic Filter

For each dimension $d \in D$ and direction $dir \in \{\vee, \wedge\}$, we now have a linear function to approximate all $(\lambda, P(o, t_i, t_j, d, dim, \lambda))$. However, using this line directly, may violate the conservativeness property, since the true function may have any monotonic increasing form, and thus, for a value λ_Q located in between two values λ_1 and λ_2 ($\lambda_1, \lambda_2 \in \Lambda, \lambda_1 < \lambda_Q < \lambda_2$) the probability is bounded by $P(\lambda_1) \leq P(\lambda_Q) \leq P(\lambda_2)$. To avoid this problem, we can exploit that the catalogue Λ is the same for all diamonds, dimensions and directions. Thus, we chose the function $f(\lambda) = l(\lfloor\lambda\rfloor)$, where $\lfloor\lambda\rfloor$ denotes the largest element of Λ such that $\lfloor\lambda\rfloor \leq \lambda$. In our running example, the function $f(\lambda)$ is depicted in Figure 15.4(c). In this example, assume that we have computed an optimal value λ_{opt} in the previous steps. The corresponding conservative approximation $f(\lambda_{opt})$ is shown.

Now, we show how these probability bounds can be used to bound the probability that an object (i.e. its corresponding chain of diamonds) satisfies the query predicate. This is done by probing each uncertain trajectory approximation (each necklace) on the query region $Q^{\Box}$. Obviously, we only have to take into account diamonds intersecting the query time range $T^{\Box}$. In turn, when probing an uncertain trajectory approximation $\diamond(o, t_i, t_j)$ on the query range $Q^{\Box}$, we only have to take the time range $[t_i, t_j]$ into account; i.e., if the time range $T^{\Box}$ of the query spans beyond $[t_i, t_j]$, we truncate $T^{\Box}$ accordingly. Consequently, in the case where more than one diamonds of an object intersect $T^{\Box}$, we can split $Q^{\Box}$ at the time dimension and separately probe the object diamonds on the corresponding query parts. The resulting probabilities obtained for individual diamonds can be treated as independent as shown by the following lemma.

Lemma 41. *Let $\diamond(o, t_i, t_j)$, $\diamond(o, t_j, t_k)$ be two successive diamonds of object o and $\diamond_1 := \diamond(o, t_i, t_j, d_1, dir_1, \lambda_1)$, $\diamond_2 := \diamond(o, t_j, t_k, d_2, dir_2, \lambda_2)$ be probabilistic subdiamonds, associated with respective probabilities $P(\diamond_1)$ and $P(\diamond_2)$ that o intersects these subdiamonds. Then, the probability $P(\diamond_1 \wedge \diamond_2)$ that o intersects both subdiamonds, is given by*

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2))$$

Proof. We first rewrite $P(inside(o, \diamond_1) \wedge inside(o, \diamond_2))$ using conditional probabilities.

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2) | inside(o, \diamond_1))$$

Furthermore, we exploit the knowledge that object o is at the observed location $o(t_j)$ at time t_j

$$P(inside(o, \diamond_1) \wedge inside(o, \diamond_2)) =$$

$$P(inside(o, \diamond_1)) \cdot P(inside(o, \diamond_2) | inside(o, \diamond_1) \wedge o(t_j))$$

Based on the Markov model assumption, we know that, given the position at t_j, the behavior of o in the time interval $[t_j, t_k]$ is independent of any position at times $t < t_j$. Thus, we obtain:

$$P(\diamond_1 \wedge \diamond_2) = P(\diamond_1) \cdot P(\diamond_2 | o(t_j))$$

Finally, the lemma is proved based on the fact that the position $o(t_j)$ has been observed, and thus, is not a random variable. □

Lemma 41 shows that the random events of two successive probabilistic diamonds of the same object are conditionally independent, given the observation in between them. Based on this, It is easy to show inductively that all pairs of random events of probabilistic diamonds of the same object are stochastically independent.

This observation allows us to compute the probability $P^{\exists}(o)$ that the whole chain of diamonds of o intersects a query window $Q^{\Box}$. Therefore, let $\{t_i, t_j\} \subseteq_{seq} o.T_{obs}$ denote the set of subsequent observations in the list of observations $o.T_{obs}$ of o.

$$P^{\exists}(o) = P(\bigvee_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} sometimes(o, t_i, t_j, Q^{\Box})$$

That is, o satisfies a PST$\tau\exists$ query, if and only if at least one diamond of o intersects $Q^{\Box}$ at least once. Rewriting yields

$$P^{\exists}(o) = 1 - P(\bigwedge_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} never(o, t_i, t_j, Q^{\Box}).$$

Exploiting Lemma 41 yields

$$P^{\exists}(o) = 1 - \prod_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} P(never(o, t_i, t_j, Q^{\Box}))$$

Using our probability bounds derived in Section 15.1.4, we obtain

$$P^{\exists}(o) \leq 1 - \prod_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} P_{LB}(never(o, t_i, t_j, Q^{\Box}))$$

which can be used to prune o, if

$$1 - \prod_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} P_{LB}(never(o, t_i, t_j, Q^{\Box})) < \tau \tag{15.4}$$

Analogously, we can prune an object o for a PST$\tau\forall$ query if

$$P^{\exists}(o) \leq 1 - \prod_{\{t_i,t_j\} \subseteq_{seq} o.T_{obs}} P_{LB}(sometimes\_not(o, t_i, t_j, Q^{\Box})) < \tau \tag{15.5}$$

If Equation 15.4 (Equation 15.5) cannot be applied for pruning, we propose to iteratively refine single diamonds of o.[6] Thus, the exact probability $P(sometimes(o, t_i, t_j, Q^{\Box}))$

[6] Heuristics to determine the order in which diamonds are refined are out of scope of this work.

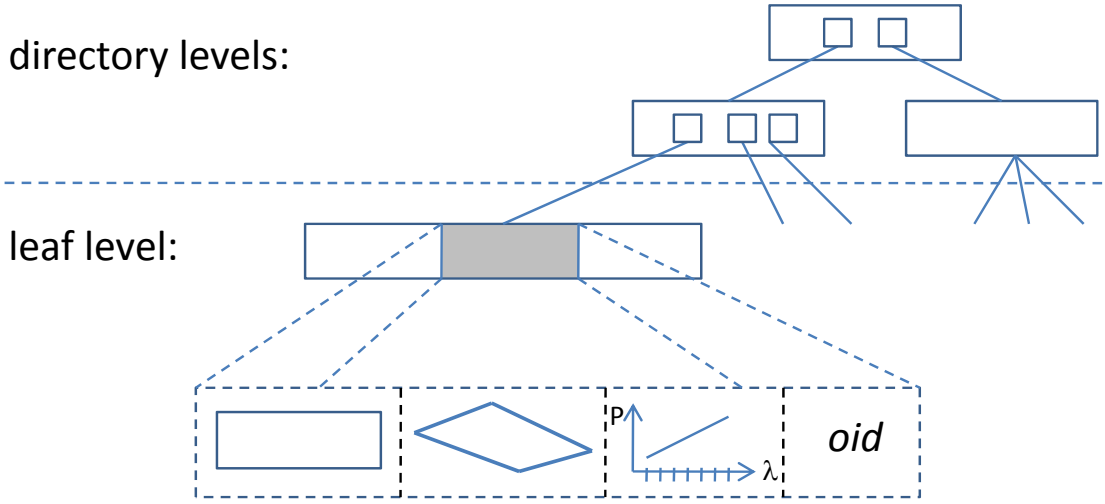


Figure 15.5: The UST-Tree.

$(P(always(o, t_i, t_j, Q^{\square})))$ is computed using the technique proposed in [66]. This exact probability of a single diamond can then be used to re-apply the pruning criterion of Equation 15.4 (Equation 15.5), by using the true probability as lower bound. When all diamonds of o have been refined, Equation 15.4 (Equation 15.5) yields the exact probability $P^{\exists}(o)$ $(P^{\forall}(o))$.

15.2 The UST-Tree

In the previous section, we showed that we can precompute a set of approximations for each object, which can be progressively used to prune an object during query evaluation. In this section, we introduce the UST-Tree, which is an R-tree-based hierarchical index structure, designed to organize the object approximations and efficiently prune objects that may not possibly qualify the query; for the remaining objects the query is directly verified based on their Markov models, as described in [66] (*refinement step*). Section 15.2.1 describes the structure of the UST-Tree and Section 15.2.2 presents a generic query processing algorithm for answering both $PST\tau\exists Q$ and $PST\tau\forall Q$ probabilistic query types efficiently.

15.2.1 Architecture

The UST-tree index is a hierarchical disk-based index. The basic structure is illustrated in Figure 15.5. An entry on the leaf level corresponds to an approximation of an object o represented by a quadruple $(\square(o, t_i, t_j), \diamond(o, t_i, t_j), \{f_{d,dir} : d \in D, dir \in \{\vee, \wedge\}\}, oid)$, containing (i) the MBR approximation $\square(o, t_i, t_j)$ (cf. Section 15.1.1), (ii) the diamond approximation $\diamond(o, t_i, t_j)$ (cf. Section 15.1.1), (iii) a set $\{f_{d,dir} : d \in D, dir \in \{\vee, \wedge\}\}$ of $2 \cdot D$ linear approximation functions for the precomputed probabilistic diamonds of o (cf. Section 15.1.5), and (iv) a pointer *oid* to the exact uncertain spatio-temporal object

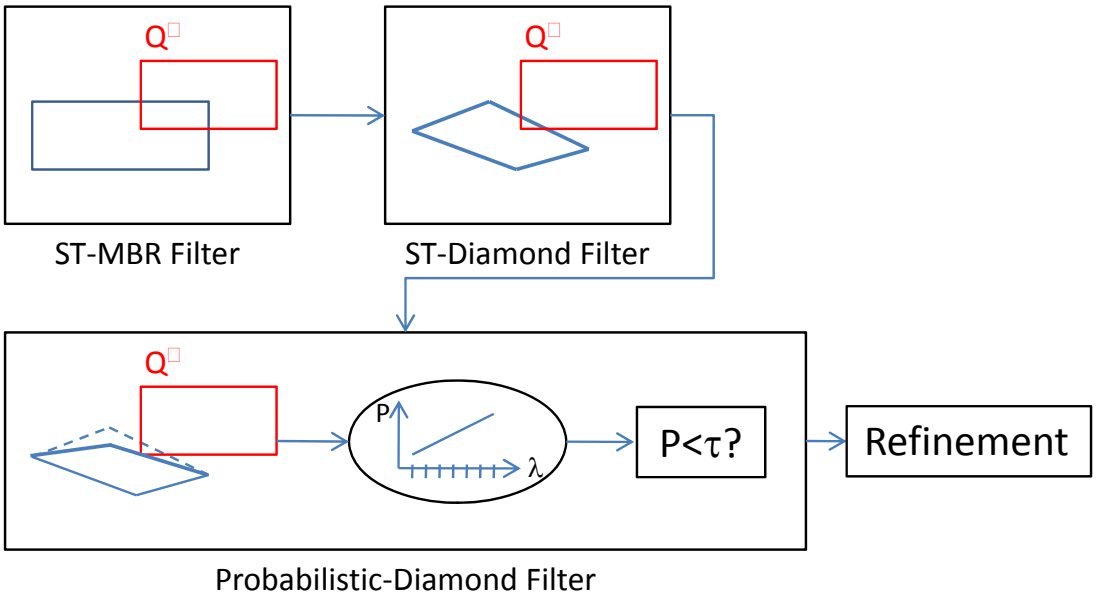


Figure 15.6: Filter-Refinement Pipeline.

description (raw object data). Intermediate node entries of the UST-tree have exactly the same structure as in an R-tree; i.e., each entry contains a pointer referencing its child node and the MBR of all MBR approximations stored in pointed subtree. Note that the necklace of each object is decomposed into diamonds, which are stored independently in the leaf nodes of the tree. Since the directory structure of the UST-tree is identical to that of the R-tree, the UST-tree uses the same methods as the R*-tree [15] to handle updates. Update operations on the UST-tree are handled in the same way as in an R*-tree. Consequently, since the structure of intermediate nodes comply with that of the R*-tree, split and merge operations on intermediate nodes are quite obvious. For the leaf level, we also adopt the split and merge heuristics of the $R^* - tree$ by just taking the *mbr*-entries into consideration.

15.2.2 Query Evaluation

Given a spatio-temporal query window $Q^{\square}$, the UST-tree is hierarchically traversed starting from the root, recursively visiting entries whose MBRs intersect $Q^{\square}$; i.e., the subtree of an intermediate entry e is pruned if $e.mbr \cap Q^{\square} = \emptyset$. For each leaf node entry e, we progressively use the spatio-temporal and probabilistic diamond approximations stored in e, attempting to filter the corresponding object, as illustrated in Figure 15.6.

In the spatio-temporal filter step, we first use $\square(o, t_i, t_j)$ (ST-MBR Filter) using simple rectangle intersection tests. If this filter fails, we proceed using $\diamond(o, t_i, t_j)$ (ST-Diamond filter) by performing intersection tests against $Q^{\square}$ as described in Section 15.1.2. Note that sometimes multiple leaf entries associated with an object are required to prune an object or confirm whether it is a *true hit*. Therefore, candidates are stored in a list until all their

diamond approximations have been evaluated.

Finally, for the remaining candidates we exploit the probabilistic filter (Probabilistic Diamond Filter) as described in Section 15.1.6. Thereby, we use the linear approximation functions $\{f_{d,dir} : d \in D, dir \in \{\vee, \wedge\}\}$ stored in the leaf-node entry in order to derive an upper bound of the qualification probability $P(\exists t \in (T^{\square} \cap [t_i, t_j]) : o(t) \in S^{\square})$ or $P(\forall t \in (T^{\square} \cap [t_i, t_j]) : o(t) \in S^{\square})$ (depending on the query predicate). For each object o which is not pruned (or reported as true hit), we accumulate in a list $L(o)$ all upper bounds of its qualification probabilities from the leaf entries that index the diamonds of o. After collecting all candidate objects, the qualification probabilities stored in the list $L(o)$ for each candidate o, are aggregated in order to derive the upper bound of the overall qualification probability for o: $P(\exists t \in T^{\square} : o(t) \in S^{\square})$ or $P(\forall t \in T^{\square} : o(t) \in S^{\square})$, respectively as described in Section 15.1.6. If this probability falls below τ we can skip o, otherwise we have to refine o by accessing the exact object data referenced by *oid*.

15.3 Conclusions

In this work, we proposed the UST-Tree which is an index structure for uncertain spatio-temporal data. The UST-Tree adopts and incorporates state-of-the art techniques from several fields of research in order to cope with the complexity of the data. We showed how the most common query types (spatio-temporal $\exists$- and $\forall$-window queries) can be efficiently processed using probabilistic bounds which are computed during index construction. To the best of our knowledge, this is the first approach that supports query evaluation on very large uncertain spatio-temporal databases, adhering to possible worlds semantics. Outside the scope of this work is the consideration of an object's location before its first and after its last observation. In both cases, the resulting diamond approximation would be unbounded. An approach to solve this problem is to define a maximum time horizon for which diamond approximations are computed. Beyond this horizon, we can use the stationary distribution of the model M to infer the location of an object.

Chapter 16

Universal Sampling of Uncertain Spatio-Temporal Data

As discussed in Part II, general query processing on uncertain spatial data is a $\#P$-hard problem, due to the exponentially large space of possible worlds. In the case of uncertain spatio-temporal data, the number of possible alternatives of a single objects, i.e., the number of alternatives routes an object may take, is already exponentially large. Despite this double exponential set of possible worlds in spatio-temporal data, it is still possible to answer some queries efficiently, as we have seen in Chapter 13 for the case of spatio-temporal window queries. Yet, there exists many more relevant spatio-temporal queries, such as nearest-neighbor queries on spatio-temporal data, ranking queries on spatio-temporal data, and many more. For these queries, it is not trivial to find an efficient solution to compute exact result probabilities. Neither it is clear if such a solution exists. Yet, in most applications requiring such queries, an approximate answer may be sufficient. For this purpose, this chapter propose a sampling approach, tailored to uncertain spatio-temporal data. It shows that a traditional (naïve) sampling approach is not applicable for spatio-temporal data, as it does not account for all observations of an object, creating a very large number of sample paths, which are not possible given all observations. To tackle this issue, we envision a model learning approach, that incorporates information about observations directly into the Markov model, following a forward-backward paradigm. The result model will allow an adapted sampling approach, that utilizes not only the information given by the initial Markov model, but also knowledge about observations. This sampling approach has the potential to efficiently an approximation of the result of any query on spatio-temporal data. This chapter presents the theoretic foundation of this sampling approach.

This approach uses Bayesian inference in order to iteratively adapt the parameters of the model, given each observation. Based on the resulting adapted model, a time inhomogeneous Markov chain, we can perform traditional sampling in order to ensure samples that are consistent with all observations, and therefore guarantee that the probability of drawing each sample is equal to the true probability of the corresponding trajectory.

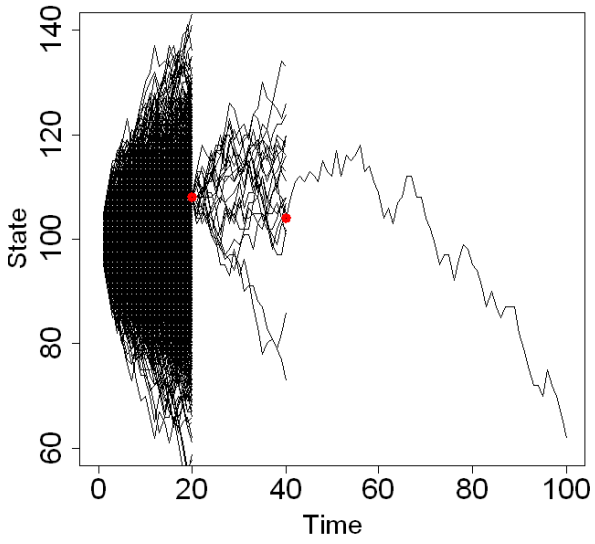


Figure 16.1: Traditional MC-Sampling.

16.1 Traditional Sampling

To sample possible trajectories of an object, a traditional Monte-Carlo approach starts by taking the first observation of an object, then transitioning to new states according to the distribution given by the underlying transition matrix. This approach however, cannot directly account for additional observations, as illustrated in Figure 16.1. Here, we assume (for simplicity) a one dimensional space, and an object randomly transitioning to adjacent states at each point of time. In the plot, 1000 samples are initiated at the first observation at time $t = 0$ and transition according to the model. Given the second observation at time $t = 20$, a number of trajectories become inconsistent (i.e., impossible given this second observation), because these trajectories do not conform to both the observation and the motion constraints of the object. Such impossible trajectories are no longer expanded to further states in Figure 16.1. At time $t = 40$, even more trajectories become invalid; in the end, only one out of a thousand samples remains possible and useful.

Clearly, the number of sample trajectories required to obtain a single valid trajectory increases exponentially in the number of observations of an object, making this traditional Monte-Carlo approach inapplicable. In the next section, we will show how to obtain possible samples efficiently, for an arbitrary number of observations. The main challenge here is not only to ensure that only possible trajectories are sampled, but also that the probability of a sampled path corresponds to the true probability of the object taking this path, given the initial model and all observations.

16.2 Adapting the Model to Observations

In the following, let $\Theta^o = \{< t_1^o, \theta_1^o >, ..., < t_m^o, \theta_m^o >\}$ denote the set of observations of a spatio-temporal object o. Each observation $< t_i^o, \theta_i^o >$ of o is a pair, containing the time t_i^o at which o has been observed in state θ_i^o. This set is assumed to be sorted, i.e., $j > i \Rightarrow t_j^o > t^o$. The traditional sampling approach uses uses the transition probabilities $P(o(t+1) = s_j | o(t) = s_i)$ given by the Markov chain to create sample trajectories. To incorporate the knowledge given by a set of observations Θ^o of an object o, we need to consider the probability

$$P(o(t+1) = s_j | o(t) = s_i, \Theta^o),$$

that is the probability that object o transitions to state s_j from state s_i at time t, given all observations. The Markov property yields:

$$P(o(t+1) = s_j | o(t) = s_i, \{\theta_i^o | t_i \geq t\})$$

This probability can be computed by assuming that o is located at s_i at time t, and then computing the probability of visiting state s_j at time $t+1$. This can be done by using an $\exists$-window query as defined in Chapter 13. An $\exists$-window query returns the probability of intersecting a set of $(time, location)$ pairs (in this case this set contains only one pair $(t+1, s_j)$), given observations in the future. Performing the above computation for each time t between the first and the last observation of o, and for each state pair s_i,s_j yields a new Markov chain, which is adapted to Θ^o. The resulting inhomogeneous Markov chain can be used to draw sample trajectories, which are guaranteed to comply with all observations, and whose probability of drawing this sample corresponds to the true probability of this trajectory given information about the initial Markov chain and all observations.Although correct and computable in polynomial time (unlike the naïve sampling approach), this approach still suffers from high computational cost for the construction of the new transition matrices: for each point in time, each entry of the original Markov chains has to be considered and a window query, which runs in $O(|S|^2 \cdot \Delta t)$ (c.f. Chapter 13), has to be performed. Although this has to be done only once, independent of the number of samples, the total time complexity of $O(|S|^4 \cdot \Delta t^2)$ makes this approach inapplicable for real applications, even when sparse matrix and vector operations are exploited.

We now present a different algorithm, which can compute the probabilities $P(o(t+1) = s_j | o(t) = s_i, \Theta^o)$ more efficiently.

16.2.1 Efficient Model Adaption

In a nutshell, this problem can be solved in the well-known forward-backward manner: Starting at the time of the first observation t_1^o with the initial observation θ_1^o, we perform transitions of object o using the original Markov chain of o until the final observation at time $t_{|\Theta^o|}$ is reached. During this *Forward*-run, Bayesian inference is used to construct a time-reversed Markov-model R_t^o of o at time t given observations in the past, i.e., a model

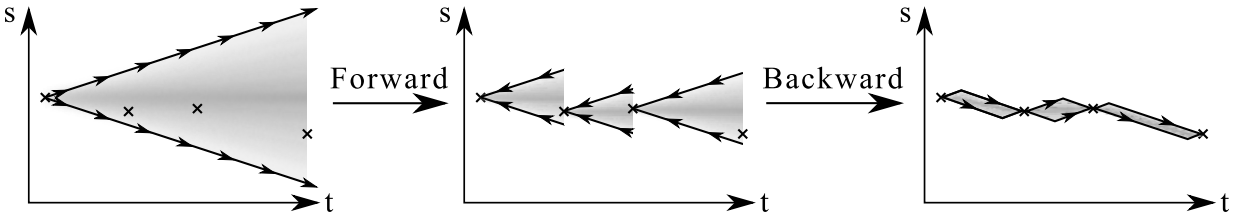


Figure 16.2: An overview over our forward-backward-algorithm.

that describes the probability

$$R^o_{ij}(t) := P(o(t-1) = s_j | o(t) = s_i, \{\theta^o_i | t^o_i < t)\})$$

of coming from a state s_j at time $t-1$, given being at state s_i at time t and given the observations in the past.

Then, in a second step, the *Backward*-run, we traverse time backwards, from time $t_{|\Theta^o|}$ to t_1, by employing the time-reversed Markov-model $R^o(t)$ constructed in the forward step. Again, Bayesian inference is used to construct a new Markov model $F^o(t-1)$ that is further adapted to incorporate knowledge about observations in the future. This new Markov model contains the transition probabilities

$$F^o_{ij}(t-1) := P(o(t) = s_j | o(t-1) = s_i, \Theta^o). \tag{16.1}$$

for each point of time t, given all observations, i.e., in the past, the present and the future.

As an example, Figure 16.2 visualizes a one-dimensional uniform random walk, i.e. a very simple Markov chain where the object may move to adjacent states with a uniform distribution. Figure 16.2(a) shows the initial model, using knowledge about the first observation only. In this case, a large set of (*time*,*location*) pairs can be reached with a probability greater than zero. The shading of reachable (*time*,*location*) pairs indicates the likelihood of these pairs, assuming that a detour is less likely than a direct path.[1] The adapted model after the forward phase is depicted in Figure 16.2(b), significantly reducing the space of reachable (*time*,*location*) pairs and adapting respective probabilities, thus drastically improving the model. Since the model of Figure 16.2 (b) is obtained trivially, the main contribution of this section is the backward-phase which adapts the Markov model to observations in the future. This task is not trivial, since the Markov-property does not hold for the future, i.e., the past is *not* conditionally independent of the future given the present. During the backward phase, we traverse the Markov chain backwards, from time $t_{|\Theta^o|}$ to t_1, by employing the information acquired in the forward-phase. This phase yields the final transition matrices for each point in time, such that all observations Θ^o are taken into account for the adapted model. The resulting final transition matrices $F^o(t-1)$ contain the transition probabilities $F^o(t-1)_{ij} = P(o(t) = s_j | o(t-1) = s_i, \Theta^o)$. Figure 16.2(c) shows the resulting final model after the backward phase. Before describing the algorithm in more detail, we prove the following lemma of the Bayes Theorem.

[1] This assumption is only made for illustration of this example. In general, a detour via a highway may be more likely than a direct path through a lake. These likelihoods are captured by the given Markov-model.

Lemma 42 (Conditional Bayes)**.**

$$P(A|B,C) = \frac{P(B|A,C)P(A|C)}{P(B|C)}$$

Proof. By applying the Multiplication Theorem of Probability and due to the commutativity of the conjunction of random events we get:

$$P(A \wedge B \wedge C) = P(C)P(B|C)P(A|B,C)$$

$$\Leftrightarrow \frac{P(A \wedge B \wedge C)}{P(B|C)P(C)} = P(A|B,C)$$

$$\Leftrightarrow \frac{P(A \wedge B \wedge C)}{P(C)} = P(A|C)P(B|A,C)$$

Mutual substitution leads to Lemma 42. □

16.2.2 Forward-Phase

The main challenge of the forward-phase is to construct necessary data structures for efficient implementation of the backward-phase, namely the time-reversed transition matrix $R^o(t)$ that summarizes the transition probabilities for a transition from time t to time $t-1$. This matrix is used to incorporate information about future observations in the backward phase.

To obtain $R^o(t)$, we can apply the theorem of Bayes as follows:

$$R^o(t)_{ij} := P(o(t-1) = s_j | o(t) = s_i) = \tag{16.2}$$
$$\frac{P(o(t) = s_i | o(t-1) = s_j) \cdot P(o(t-1) = s_j)}{P(o(t) = s_i)}$$

By assuming existence of all observation $past^o(t) := \{\theta_i^o | t_i^o < t)\}$ of o that occurred before time t, all events become further conditioned to these observations as follows, using Lemma 42:

$$R^o(t)_{ij} := P(o(t-1) = s_j | o(t) = s_i, past^o(t)) = \tag{16.3}$$
$$\frac{P(o(t) = s_i | o(t-1) = s_j, past^o(t)) \cdot P(o(t-1) = s_j | past^o(t))}{P(o(t) = s_i | past^o(t))}$$

The probability $P(o(t) = s_i | o(t-1) = s_j, past^o(t))$ can be rewritten as $P(o(t) = s_i | o(t-1) = s_j)$, exploiting the Markov property (Eq. 56). This probability is given by the original Markov-chain $T^o(t)$.

Furthermore, both priors $P(o(t-1) = s_j | past^o(t))$ and $P(o(t) = s_i | past^o(t))$ can be computed in a single run: We start at $t = t_1$ using the initial distribution of θ_1^o. Then, transitions are performed iteratively using the original Markov chain $T^o(t)$. For each intermediate point of time t, all probabilities $P(o(t-1) = s_j | past^o(t))$ are memorized.

In each iteration of the forward algorithm, where a new observation $present^o(t) := \theta_x^o, < t_x^o, \theta_x^o > \in \Theta^o, t_x^o = t$ is reached, we incorporate this information to the model. Therefore, we compute $P(o(t) = s_i | past^o(t), present^o(t))$ from $P(o(t-1) = s_j | past^o(t))$ for all states s_i, employing all s_j. This is done by exploiting the assumption that observations are mutually *independent*, i.e. that the error made between two measurements is independent. This allows to compute the probability of the event $Y_j := [o(t) = s_j | past^o(t)] =$ "o is in state s_j at time t given observations before t", *and* the event $Z_i := [o(t) = s_i | present^o(t)] =$"$o$ is in state s_i given the observation θ_x^o at time t" exploiting

$$P(Y_j \wedge Z_i) = P(Y_j) \cdot P(Z_i). \tag{16.4}$$

Clearly, the above joint distribution between states observed due to observations before t and states observed due to the observation at time t allows contradicting observations where both random variables result in different states. Let $Y := [o(t) | past^o(t)]$ and $Z := [o(t) | present^o(t)]$. Then such contradicting worlds can be removed by conditioning the probability of Equation 16.4 events to the event $Y = Z$.

$$P\left(Y_j \wedge Z_i | Y = Z\right) = \frac{P(Y = Z | Y_j \wedge Z_i) \cdot P(Y_j \wedge Z_i)}{P(Y = Z)} \tag{16.5}$$

In the above equation, the term $P(Y = Z | Y_j \wedge Z_i)$ acts as an indicator function that is one if for the two random variables Y and Z it holds that $Y = Z$, i.e., if the observations are non-contradicting, and zero otherwise. The term $P(Y_j \wedge Z_i)$ can be rewritten to the product of the probabilities of both observations to materialize to s_j according to Equation 16.4. Finally, the denominator $P(Y = Z)$ corresponds to the total probability that both observations are non-contradicting, and can rewritten as $\sum_i Y_i \cdot Z_i$ also exploiting independence of observations in Equation 16.4.

In summary, the total probability $P(Y_j \wedge Z_i | Y = Z) = P([(o(t) = s_j | past^0(t)] \wedge [o(t) = s_i | present^o(t))] | s_i = s_j)$ that object o is at state s_j at time t can be computed by performing a simple element-wise multiplication between the j'th element of vector $P(o(t) = s_j | past^o(t))(j)$ and the j'th element of vector θ_t^o.

Now that all observations at time t and before are considered, both data structures, the new state vector $\vec{s}^o(t)$ and the adapted transition matrix $R^o(t)$ can be efficiently derived from the following temporary matrix, computed in Line 4 of Algorithm 11:

$$X'(t) = T^o(t-1)^T \cdot diag(\vec{s}^o(t-1))$$

The equation is equivalent to a simple transition at time t, except that the state vector is converted to a diagonal matrix first. This trick allows to obtain a matrix describing the distribution of the position of o at time $t-1$ and t, instead of a simple distribution of the location of o at time t. Formally, each entry $X'(t)_{i,j}$ corresponds to the probability $P(o(t-1) = s_j \wedge o(t) = s_i | past^o(t))$ which is equivalent to the *numerator* of Eq. 16.3.[2] To

[2]The proof for this transformation $P(A \cap B | C) = P(A|C) \cdot P(B|A, C)$ can be derived analogously to Lemma 42.

Algorithm 11 AdaptTransitionMatrices(o)

1: {Forward-Phase}
2: $\vec{s^o}(t_1^o) = \theta_1^o$
3: **for** $t = t_1^o + 1;\ t \leq t^o_{|\Theta^o|};\ t{+}{+}$ **do**
4: $\quad X'(t) = T^o(t-1)^T \cdot diag(\vec{s^o}(t-1))$
5: $\quad \vec{s^o}(t)_i = \sum_{j=1}^{|S|} X'_{ij}(t)$
6: $\quad R^o(t)_{ij} = \frac{X'_{ij}(t)}{\vec{s^o}(t)_i}$
7: $\quad$ {Incorporate observation}
8: $\quad \vec{s^o}(t)_i = normalize(\vec{s^o}(t) \bullet present(t))$
9: **end for**
10: {Backward-Phase}
11: **for** $t = t^o_{|\Theta^o|} - 1; t \geq t_1^o; t$-- **do**
12: $\quad X'(t) = R^o(t+1)^T \cdot diag(\vec{s^o}(t+1))$
13: $\quad \vec{s^o}(t)_i = \sum_{j=1}^{|S|} X'_{ij}(t)$
14: $\quad F^o(t)_{ij} = \frac{X'_{ij}(t)}{\vec{s^o}(t)_i}$
15: **end for**
16: {Return modified object; state vectors and changed transition matrices are relevant}
17: **return** o

obtain the denominator of Eq. 16.3 we first compute the row-wise sum of $X'(t)$ in Line 5:

$$\forall i \in \{1..|S|\} : \vec{s^o}(t)_i = \sum_{j=1}^{|S|} X'(t)_{ij}$$

The resulting vector directly corresponds to $\vec{s^o}(t)$, since for any matrix A and vector x it holds that $A \cdot x = rowsum(A \cdot diag(x))$. By employing this rowsum operation, only one matrix multiplication is required for computing $R^o(t)$ and $\vec{s^o}(t)$.

Next, the elements of the temporary matrix $X'(t)$ and the elements of $o.\vec{s}(t)$ can now be normalized in Equation 16.3, as shown in Line 6 of the algorithm:

$$\forall i, j \in \{1..|S|\} : R^o(t)_{ij} = \frac{X'(t)_{ij}}{\vec{s^o}(t)_i}$$

Here, the resulting reverse probabilities of R are stored directly in the matrix $T^o(t)$, so matrix T^o can be discarded. This optimization allows to reuse the allocated space of matrix $T^o(t)$, since the old transition probabilities are no longer used in the remainder of the algorithm. Finally, possible observations at time t are integrated in Line 8, using Eq. 16.5.

Backward Phase

During the backward phase, we traverse time backwards, to propagate information about future observation back to past points of time, as depicted in Figure 16.2(c), thus terminating our algorithm. This phase is equivalent to the Forward-Phase, except that we do not use the initial matrices but rather the reverse transition matrices $R^o(t)$ created during the forward phase, which contain, for each point of time t, transition probabilities already conditioned to observations at time t and before time t. The main benefit of $R^o(t)$ is to allow transitions backwards in time, since $R^o(t)$ is defined as a transition matrix containing the probabilities of going from one state at time $t+1$ to another state at time t. In a nutshell, $R^o(t)$ is a Markov chain with the time axis being reversed. The following reverse Markov property holds for each element R^o_{ij} of matrix R^o:

$$P(o(t) = s_j | o(t+1) = s_i, o(t+2) = s_{t+2}, ..., o(t+k) = s_{t+k}) =$$

$$P(o(t) = s_j | o(t+1) = s_i) \tag{16.6}$$

As an initial state for the backward phase, we use the state vector $\vec{s^o}(t^o_{|\Theta^o|})$ that results from Section 16.2.2, by element-wise multiplication of the vector derived by using the Markov-chain and all observation but the final observation and the vector $\theta^o_{|\Theta^o|}$ of the final observation. This way, we take the final observation as given, making any further probabilities that are being computed conditioned to this observation. At each point of time $t \in [t_{|\Theta^o|}, t_1]$ and each state $s_i \in S$, we compute the probability that o is located at state s_i at time t *given* (conditioned to the event) that the observations $future^o(t) := \{\theta^o_i | t^o_i > t)\}$ at times later than t are made. Using the probabilities of $R^o(t)$ which are already conditioned to the observations $past^o(t)$ and $present^o(t)$, this yields the final transition probabilities $F^o_{ij}(t) := P(o(t) = s_j | o(t+1) = s_i, \Theta^o)$, thus including all observations $\Theta^o = past^o(t) \cup present^o(t) \cup future^o(t)$. To compute these final probabilities, which we need for our sampling approach (c.f. Section 16.2), we once again exploit Lemma 42:

$$P(o(t+1) = s_j | o(t) = s_i, \Theta^o) =$$

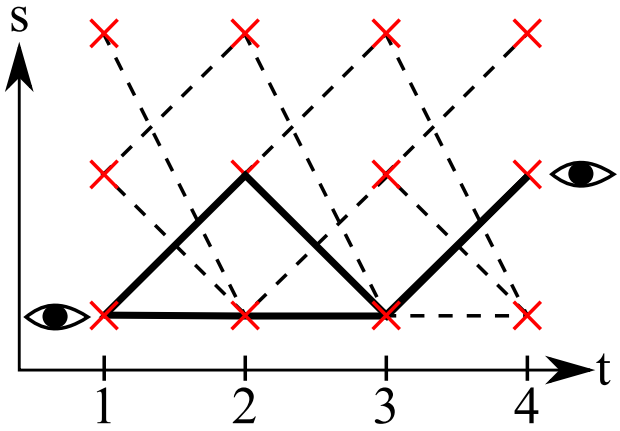
$$\frac{P(o(t) = s_i | o(t+1) = s_j, \Theta^o) \cdot P(o(t+1) = s_j | \Theta^o)}{P(o(t) = s_i | \Theta^o)}$$

By exploiting the reverse Markov property (c.f. Equation 16.6), this becomes equal to:

$$P(o(t+1) = s_j | o(t) = s_i, \Theta^o) = \frac{P_1 \cdot P(o(t+1) = s_j | \Theta^o)}{P(o(t) = s_i | \Theta^o)}, \tag{16.7}$$

$$\text{where } P_1 = P(o(t) = s_i | o(t+1) = s_j, present^o(t), past^o(t))$$

The probability P_1 is now given by the entries of matrix $R^o(t)$, by its definition. Again, both priors $P(o(t+1) = s_j | \Theta^o)$ and $P(o(t) = s_i | \Theta^o)$ can be computed in a single run: We start at $t = t_{|\Theta^o|}$ using the distribution of $\vec{s^o}(t^o_{|\Theta^o|})$. Then, transitions are performed



(a)

$$T = \begin{pmatrix} 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{4}{5} & \frac{1}{5} \end{pmatrix}$$

(b)

Figure 16.3: Exemplary Markov Chain, Visualization (a) and Transition Matrix (b).

backwards until time $t = t_1$ is reached, and for each intermediate point of time t, all probabilities $P(o(t) = s_i, \Theta_1^o)$ are memorized.

Note that even if the initial transition matrix was homogeneous (time-invariant), the resulting transition matrices would typically be inhomogeneous. Also note, that further observations will lead to a higher sparsity of the transition matrices, since new observations may render a large number of (time,state) pairs unreachable, as illustrated in Figure 16.2 (c). Finally, we further note that the transition matrices become more sparse at points of time close to observations. The formal backward algorithm can be found in Lines 11-14 of Algorithm 11 and follows the structure of the forward phase. The returned state vectors $\vec{s}^o(t)$ can be employed as an initial distribution for starting sampling at any arbitrary point in time.

The overall complexity of this algorithm is $O(\Delta t|\mathcal{S}|^2)$. The initial matrix multiplication requires $|\mathcal{S}|^2$ multiplications. While the complexity of a matrix multiplication is usually in $O(|\mathcal{S}|^3)$, the multiplication of a matrix with a diagonal matrix, i.e., $T^T \cdot s$ can be rewritten as $T_i^T \cdot s_{ii}$, which is actually a multiplication of a vector with a scalar, resulting in an overall complexity of $O(\mathcal{S}^2)$. Rediagonalization needs $|\mathcal{S}|^2$ additions as well, such as renormalizing the transition matrix, yielding $3 \cdot \Delta t \cdot |\mathcal{S}|^2$ for the forward phase. The backward phase has the same complexity as the forward phase, leading to an overall complexity of $O(\Delta t|\mathcal{S}|^2)$.

Example

To make the algorithm more clear, consider the following example from Figure 16.3(a), consinsting of four consecutive timestaps and three states. The initially time-homogeneous transition matrix, visualized as dashed lines in Figure 16.3(a), is given in Figure 16.3(b). Furthermore, let two observations $\theta_1^o = (0, 0, 1)^T$ and $\theta_2^o = (0, 1, 0)^T$ be given. Clearly, given these observations, there is a total of only two possible trajectories (the black lines): $p_1 = (s_3, s_3, s_3, s_2)$ and $p_2 = (s_3, s_2, s_3, s_2)$. The probabilities of these paths is $P(s_3, s_3, s_3, s_2) = 0.2 \cdot 0.2 \cdot 0.8 = 0.032$ and $P(s_3, s_2, s_3, s_2) = 0.8 \cdot 0.5 \cdot 0.8 = 0.32$. Thus, given the new observations, the probability of trajectory p_1 equals $\frac{0.032}{0.032+0.32} = \frac{1}{11}$ and the

probability of trajectory is $\frac{0.32}{0.032+0.32} = \frac{10}{11}$. Albeit correct, it is impractical to compute this exact result in practise, as it requires to enumerate the exponentially large set of all possible trajectories. Furthermore, as discussed in this Section, a naive sampling approach is not viable, as in practice, only an exponentially small number of samples will hit all observations. Thus, we show in the following how to adapt the Markov model of this example, to allow drawing samples that are guaranteed to draw samples conform to the observations, without incurring any bias of the distributions of these correct samples.

To generate the adapted transition matrices that can only produce the two valid trajectories denoted within the picture, we run the algorithm introduced above.

Forward. First, we multiply the transposed transition matrix T with the initial diagonalized observation:

$$X'(2) = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{4}{5} \\ 1 & \frac{1}{2} & \frac{1}{5} \end{pmatrix} \cdot \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{4}{5} \\ 0 & 0 & \frac{1}{5} \end{pmatrix}$$

$$\Rightarrow \vec{s}(2) = \begin{pmatrix} 0 \\ \frac{4}{5} \\ \frac{1}{5} \end{pmatrix}, R(2) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

The vector $\vec{s}(2)$ denotes the distribution of the object after the first transition, $R(2)$ the backward transition probabilities of the object. In the following iteration, we build uppon $s(1)$ and get:

$$X'(3) = \begin{pmatrix} 0 & \frac{2}{5} & 0 \\ 0 & 0 & \frac{4}{25} \\ 0 & \frac{2}{5} & \frac{1}{25} \end{pmatrix} \Rightarrow \vec{s}(3) = \begin{pmatrix} \frac{2}{5} \\ \frac{4}{25} \\ \frac{11}{25} \end{pmatrix}, R(3) = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{10}{11} & \frac{1}{11} \end{pmatrix}$$

The next iteration works equivalently:

$$X'(4) : \begin{pmatrix} 0 & \frac{2}{25} & 0 \\ 0 & 0 & \frac{44}{125} \\ \frac{2}{5} & \frac{2}{25} & \frac{11}{125} \end{pmatrix} \Rightarrow R(4) : \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{50}{71} & \frac{10}{71} & \frac{11}{71} \end{pmatrix}, \vec{s}(4) = \begin{pmatrix} \frac{2}{25} \\ \frac{44}{125} \\ \frac{71}{125} \end{pmatrix}$$

Since at $t = 3$ we have an observation, we incorporate it by piecwise multiplication and normalization, getting $s(3) = s(3) \cdot \theta_2^o = (0, 1, 0)^T$

Backward. The backward phase is quite similar to the forward phase, however we reuse the reverse transition matrices computed during the forward phase. We get:

$$X'(3) = R(4)^T \cdot diag(s(4)) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\Rightarrow F(3) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \vec{s}(3) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

$$X'(2): \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \frac{10}{11} \\ 0 & 0 & \frac{1}{11} \end{pmatrix} \Rightarrow F(2) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{pmatrix}, \vec{s}(2) = \begin{pmatrix} 0 \\ \frac{10}{11} \\ \frac{1}{11} \end{pmatrix}$$
$$X'(1): \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{10}{11} & \frac{1}{11} \end{pmatrix} \Rightarrow F(1) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & \frac{10}{11} & \frac{1}{11} \end{pmatrix}, \vec{s}(1) = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

The matrices $T(i)$ computed during the backward phase are returned as the set of adapted, now time-inhomogeneous transition matrices.

Sampling Process

Once the transition matrices for each point of time have been adapted to incorporate knowledge about all observations, the actual sampling process is simple: For each object o, each sampling iteration starts by sampling an initial position $s^o(t_1)$ as a random realization of the random variable defined by the initial distribution of the object's first *adapted state vector* at time t_1. Then, a random walk is performed, using the transition probabilities given by the adapted transition matrices until the final observation of o at time t_m is reached: For each $t_1 < t \leq t_m$ a state $s^o(t)$ is sampled by a random transition using the adapted transition matrix $T(t_1)$. Thus, $s(t)$ is the realization of the random variable defined by the $s(t-1)$'th line of the adapted transition matrix $T^o(t)$. On these certain trajectories $s^o(t), t_1 \leq t \leq t_m$, which are realized for each object, standard algorithms for certain trajectories can be applied. This allows to efficiently answer any query on uncertain spatio-temporal data, for which efficient solutions for the case of certain trajectories ([216]) exist. To assess the quality of the approximation, techniques presented in Section 2.6 can be used.

16.3 Research Directions

This chapter has given the required theoretical foundations for an universal sampling approach, that, unlike traditional sampling approaches, allows to efficiently approximate the result of most queries on uncertain spatio-temporal data. The only requirement for this approximation to be efficient, is that an efficient solution for the certain case must exist, i.e., for the case of traditional (certain) trajectories. This opens a plethora of new applications. The vision is to extend all kinds of relevant queries ([216]) to the case of uncertain spatio-temporal data. This extension is highly useful and necessary in a domain, where uncertain is highly inherent and ubiquitous. It will fuel the vision of data-driven research, and will take a step towards the challenge presented by the recent McKinsey report ([137]) to unearth a wealth of potential revenue hidden in geo-spatial data.

Chapter 17

Experimental Evaluation

17.1 Experimental Setup

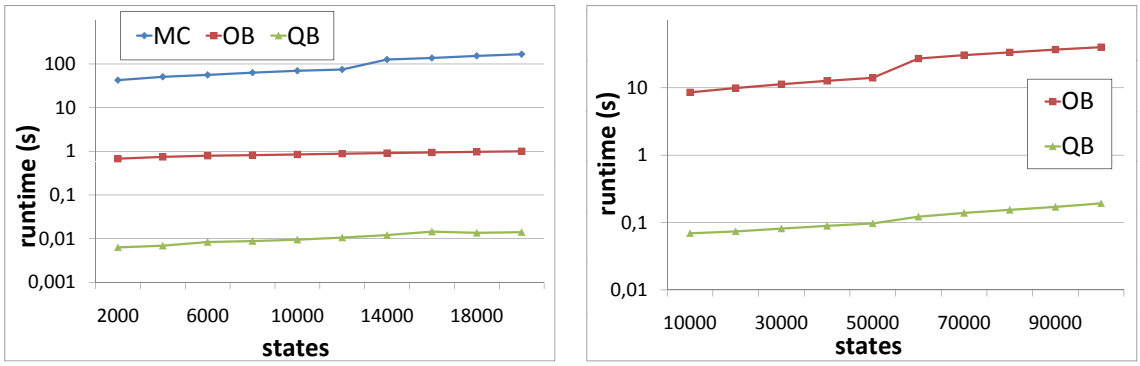
For our experimental evaluation we used synthetic as well as real datasets. In order to observe the influence of data characteristics we used several parameters for the construction of the *synthetic datasets*. Each experiment is performed using a database of $|\mathcal{DB}|$ objects. The location of each object at time t_0 is given by a PDF over a certain number of states. This value is controlled by the parameter *object_spread* and characterizes the amount of uncertainty in the database. The total number of states of the system is characterized by $|\mathcal{S}|$. To control the density of the transition matrix (which corresponds to the number of possible transitions in the modeled system) we used the parameter *state_spread*. From each state it is possible to transition into *state_spread* states. To model locality of the transitions within the system we also introduced a parameter which bounds the possible states which can be reached by one transition. An object in state s_i can only transition into states $s_j \in [s_{i-max\_step/2}; s_{i+max\_step/2}]$. All parameters for the synthetic datasets are summarized in Table 17.1.

As *first datasets* we used two road network datasets. The first is the road network of North America which consists of 175,813 nodes and 179,102 edges. As this is a rather sparse graph we also extracted the road network from Munich which has 73,120 nodes and 93,925 edges. The transition matrix is equivalent to the adjacency matrix of the corresponding graph. This means each node is treated as a state and each edge corresponds to two non-zero entries in the transition matrix. The value of the non-zero entries of one line in the matrix are set randomly and sum up to one. In this way they reflect the transition probabilites from one node in the road network to its directly connected neighbors. While the underlying road network corresponds to a real road network, the transition probabilities are synthetic, such that this data set is considered synthetic in the remainder of this chapter.

A *second real data set* is generated from a set of GPS trajectories of taxis in the city of Beijing [208]. The data set was generated using the techniques of [47] to obtain both a set of possible states (corresponding to crossroads) and a transition matrix reflecting the possible movements of the cabs. This process yields a state space consisting of about

parameter	value range	default
$\|\mathcal{DB}\|$	1,000 - 100,000	10,000
$\|\mathcal{S}\|$	2,000 - 100,000	100,000
object_spread	5	5
state_spread	1 - 20	5
max_step	10 - 100	40

Table 17.1: Parameters for the synthetic datasets



(a) Comparison of all approaches

(b) Comparison of FMM and BMM on larger datasets

Figure 17.1: Increasing number of states

3000 states and the corresponding transition matrices and direct edges between states. We assume that a-priori, all objects utilize the same Markov model M. In this data set, both the underlying road network as well as the estimated transition probabilities are taken from real trajectories. Thus this data set will be considered a real-world data set in the remainder of this chapter.

Observations. Additionally to the positions of the states and the transition matrix we further need observations from each object in order to build a database. The observations were constructed by a directed random walk through the underlying graph (states = vertices and non-zero transitions = edges). At some time steps we memorize the current position of the object and take these time-state-tuples as an observation of the object. The time steps between two successive observations was randomly chosen from the interval [10,15] if not stated otherwise. For the observations of the real dataset we used the GPS data of taxis where the time between two signals is between 2 and 20 minutes.

17.2 Spatio-Temporal Window Queries

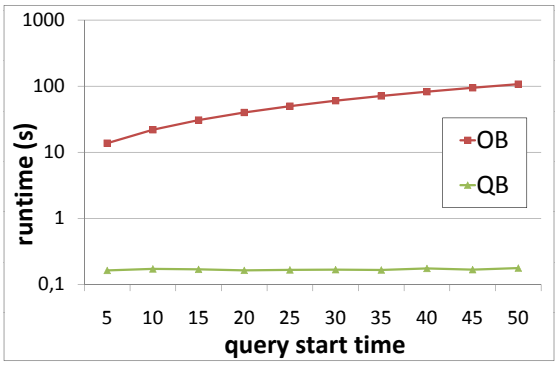
In our experiments we evaluate object-based (**OB**) and query-based (**QB**) query processing using several query predicates ($\exists, \forall$ and k-count). Additionally we compare these approaches to a Monte-Carlo based method (**MC**). The **MC** approach samples paths of each object and outputs the fraction of the sampled paths which fulfill the query predicate. Sampling the path of an object requires first drawing a start state from the objects distribution. Afterwards for each timestep a state from the successor states of the current state is chosen according to the probability distribution given by the transition matrix. Note that **MC** only returns approximate results, where the accuracy can be improved if more paths are sampled. Since the sampling of paths is equivalent to a Bernoulli sequence, the standard deviation between the sampled probability ($\hat{p}$) and the true probability (p) is given by $\sigma_{\hat{p}} = \sqrt{\frac{p \cdot (1-p)}{n}}$. For 100 samples, the standard deviation between p and $\hat{p}$ is thus at least 5% and gets worse for small and large values of p.

All experiments were run on a single 64-Bit machine with an Intel Xeon 5160 processor with 3.0 GHz and 32GB of RAM. The computations where performed using MATLAB R2011a. Unless mentioned otherwise, we generated 10,000 objects randomly distributed across the state space and the query window is defined by the states [100, 120] and time interval [20, 25]. For the Monte-Carlo based approach the number of drawn samples was set to 100.

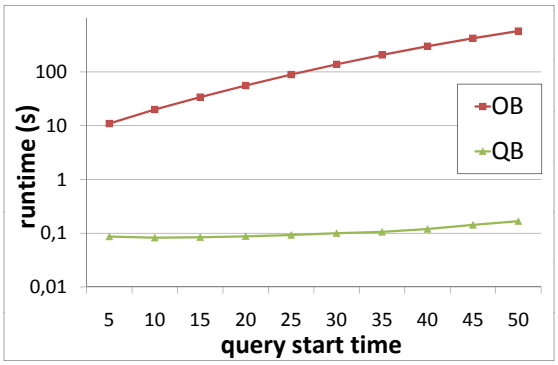
The experiments shown in Figures 17.2(a) and 17.2(b) show the dependency of the PST$\exists$Q algorithms on the timeslot we want to query on synthetic and real data sets. The runtime of **OB** is increasing much faster than the runtime of **QB**. As expected the runtimes of both algorithms suffer from a longer glance in the future as the vectors to be multiplied become less sparse with each time stamp. Besides this, **QB** should not be influenced by the number of timestamps whereas the runtime of the **OB** approach scales linearly to that parameter.

In the first experiment, we vary the size of the state space $|S|$ and measure the cost of query evaluation for a PST$\exists$Q. In Figure 17.1(a), we used a relatively small synthetic dataset ($|\mathcal{DB}| = 1,000$, $|\mathcal{S}| = [1,000; 10,000]$). The **MC** approach is computationally very demanding in comparison to the other two algorithms. The reason is, that even for such a small setting the Monte-Carlo based approach has to draw a very high amount of samples. Note that already for a small number of sampled paths (we used 100 in this setting) this approach becomes expensive, because the sampling of one path already requires to draw as many samples as the considered stamps in the query. This corresponds to 2,500 samples for one object in the database. Due to these high costs we excluded the **MC** algorithm from the remaining experiments. As expected the **QB** approach is much faster than the **OB** approach. Figure 17.1(b) shows how these two methods scale at larger datasets ($|\mathcal{DB}| = 100,000$, $|\mathcal{S}| = 10,000$).

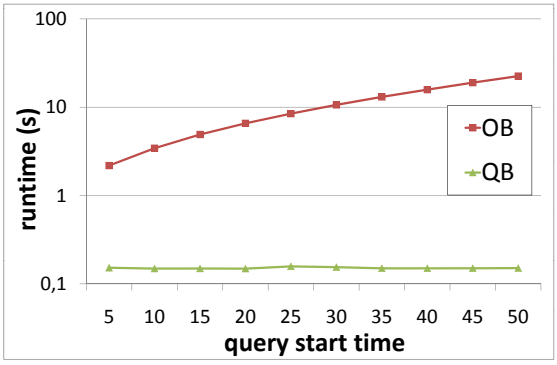
In Figure 17.3(a) we compare the three proposed query types PST$\exists$Q, PST$\forall$Q and PSTkQ query using the object-based approach. Obviously for the PSTkQ we have to maintain not only one but multiple vectors (as many as the number of times in $T^{\square}$) per



(a) synthetic data



(b) munich dataset



(c) NA dataset

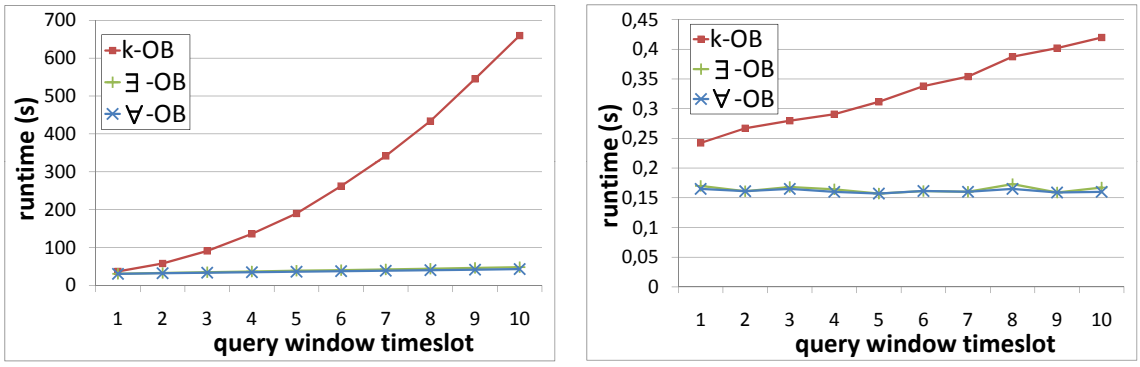
Figure 17.2: Increasing Time

object, which leads to an increased runtime. The PST∃Q and PST∀Q had equal runtime in all experimental settings. Using the **QB** approach all queries run in a fraction of a second and the runtime of PSTkQ seems to scale rather linearly with k (cf. Figure 17.3(b)).

In the next set of experiments, the runtime behavior of the two approaches w.r.t. different locality parameters is tested. Figure 17.4(a) shows the runtime for increasing the *max_step* parameter whereas Figure 17.4(b) shows results for increasing *state_spread* parameter (Note that the algorithm runtimes are marked at different axes). Both algorithms scale at most linearly with those parameters.

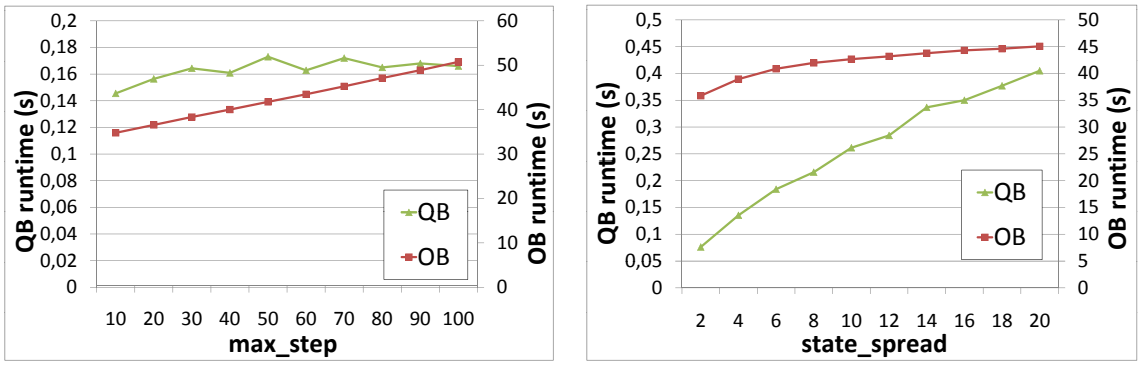
17.2.1 Impact of the UST-Tree Index

The UST-Tree was implemented in Java adapting the R*-Tree implementation of the ELKI Framework [3]. For all operations involving matrix operations (e.g., the refinement step of queries) we used MATLAB for efficient processing. The code is publicly available on our project page [1]. All query performance evaluation results are averaged over 1000 queries. The spatial extent of the query windows in each dimension was set to 0.1 and the duration of the queries was set to 10 time steps by default. Unless otherwise stated, we performed PST$\tau\exists$ queries, with $\tau = 0.5$. The page size of the tree was set to 4 KB.



(a) OB approach (b) QB approach

Figure 17.3: Three query predicates in comparison

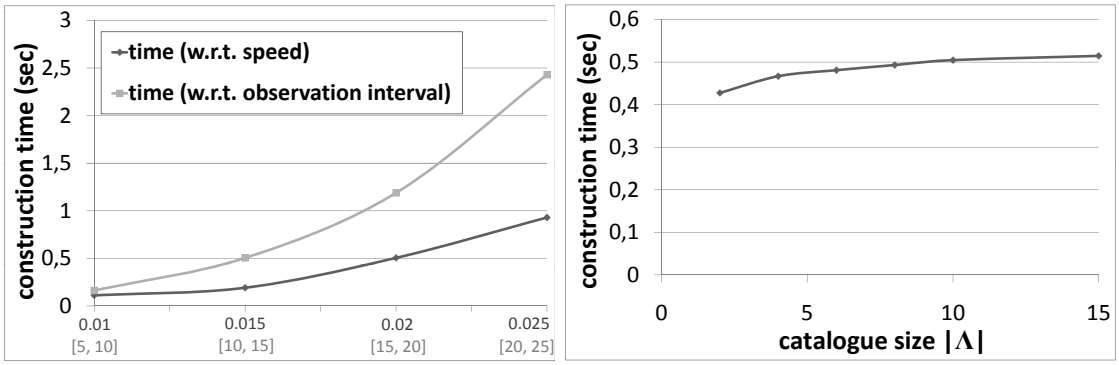


(a) Impact of max_step (b) Impact of state_spread

Figure 17.4: Comparison of QB and OB behavior with scaling parameters

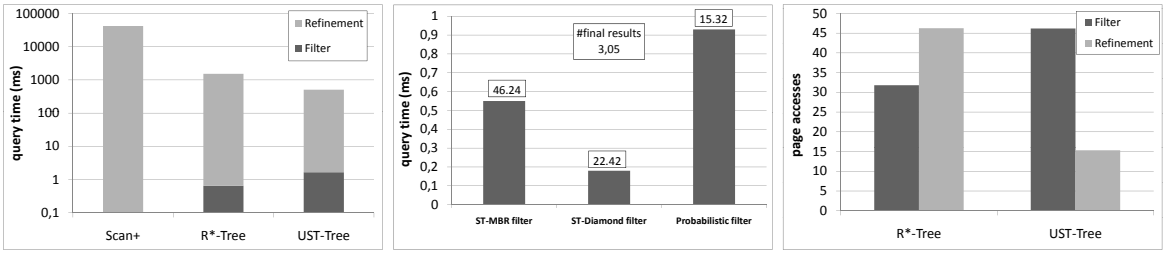
17.2.2 UST-tree Construction

The first experiment investigates the cost of index construction. In particular, we evaluate the cost for generating the spatio-temporal and probabilistic diamond approximations used to build the entries of the leaf level. This is the bottleneck of constructing and updating the UST-Tree, since actual restructuring operations of the index in our experiments always take at most 1ms. On the other hand, the construction cost of the probabilistic diamonds is usually 2-3 orders of magnitude higher, as illustrated in Figure 17.5(a). This cost is reasonable, since the construction of a probabilistic diamond is comparable to the construction of $2 \cdot D \cdot |\Lambda|$ subdiamonds, which in turn corresponds to one refinement step (considering the subdiamond as a query window). However these construction times pay off, when the query load on the database is reasonable. Figure 17.5(a) illustrates the construction time as a function of the speed of the objects and the number of time steps between successive observations. From a theoretical point of view, both parameters linearly increase the number of reachable states, i.e. the density of the sparse vectors representing the uncertain position



(a) diamond parameter (b) catalogue size

Figure 17.5: diamond construction



(a) CPU Comparison (b) CPU Time of Filter (c) Page Accesses

Figure 17.6: Overall Performance (Synthetic Data Set)

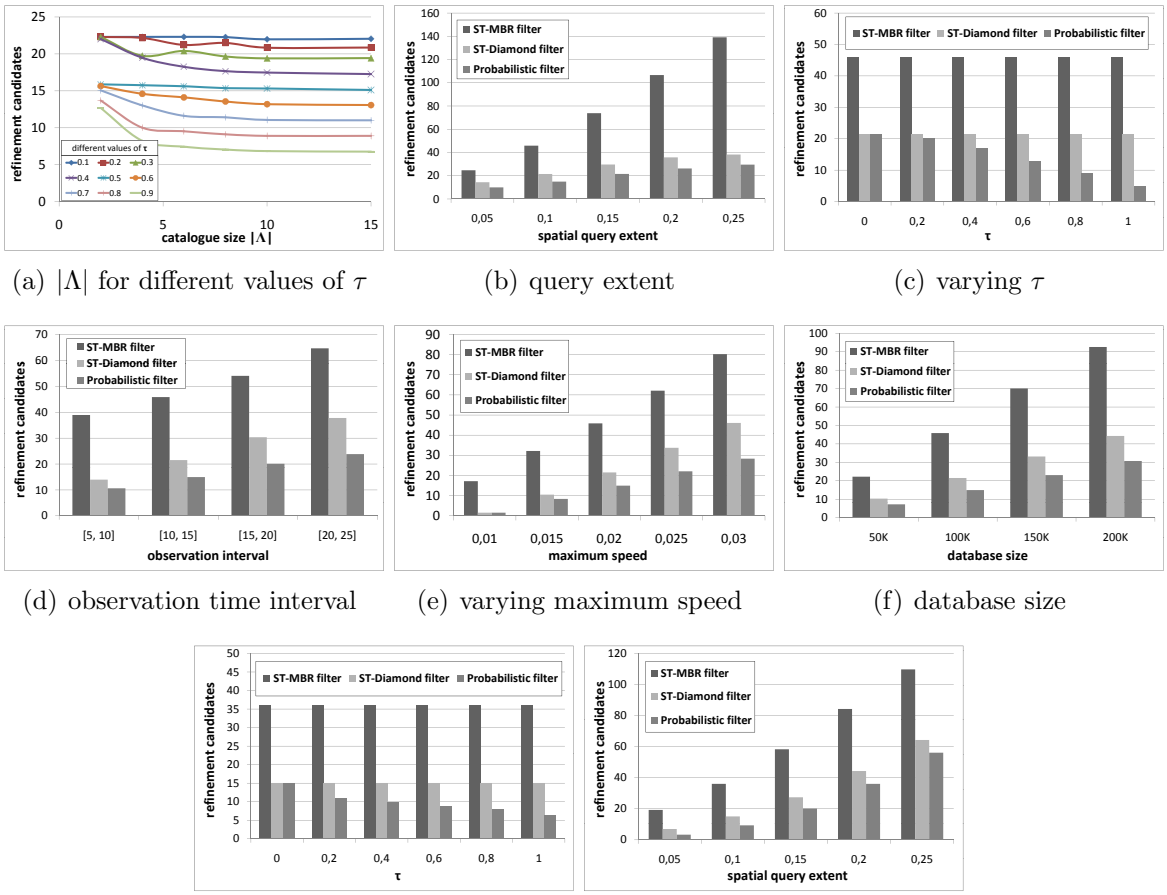
of an object at one point of time. The results reflect the theoretical considerations showing a quadratic runtime behavior with respect to both parameters. In a streaming scenario with several updates/insertions per second and large probabilistic diamonds (due to high speed of objects or large intervals between observations), the construction of probabilistic diamonds can be performed in parallel and is therefore still feasible.

Since the number of subdiamonds is determined by parameter $|\Lambda|$, we illustrate the construction time w.r.t. this parameter in Figure 17.5(b). Theoretically this parameter should have a linear impact on the construction time. However our implementation exploits the monotonicity of the uncertain trajectories regarding probabilistic subdiamonds. This means that a trajectory which is not included in the probability of a subdiamond, is also not included in larger subdiamonds in the same dimension and direction. This observation explains the sublinear runtime w.r.t. $|\Lambda|$.

17.2.3 Query Performance

Next, we evaluate the performance of our UST-tree-based query processing methods. Just like for most uncertain database queries, the bottleneck in query evaluation is the refinement step. Given two successive observations of an object o and a query window $Q^{\Box}$, refinement involves loading the Markov Chain $o.M$ of o and computing the probability of the object to satisfy the query according to Section 15.1.6. In the first set of experiments we demonstrate the overall performance of the UST-tree in comparison with two competitors (cf. Figure 17.6). The experiments have been performed on the synthetic data set. *Scan+* is a scan based query processing implementation, i.e. without employing any index [66]. Between each two successive observations of an object, refinement is performed immediately, i.e. there is no filter cost. We enhanced the implementation of *Scan+* by prepending a simple temporal filter, which does not consider observation pairs which do not temporally overlap the query window. The *R*-Tree* competitor approximates all possible locations (i.e. state-time pairs) between two successive observations of an object using only $\Box(o, t_i, t_j)$. These MBRs are then indexed using a conventional R*-Tree [15]. In Figure 17.6(a), we show the average query runtime in terms of CPU cost (I/O cost is not the bottleneck of this problem), for the three competitors. The cost are split into filter and refinement cost. Although *R*-Tree* has less filter cost, the overall query performance of the UST-tree is around 3 times better than that of *R*-Tree* (note the logarithmic scale). This is attributed to the effectiveness of the different filter steps used by the UST-tree, while the overhead of the UST-tree filter is negligible compared to the savings in refinement cost. We also evaluated the cost as well as the effectiveness of the individual filter steps of the filter-refinement pipeline used by the UST-tree; the results are shown in Figure 17.6(b). The bars show the overall runtime (query time) of each filter and the numbers within the boxes denote the effectiveness of the filter in terms of remaining (observation pair) candidates after the corresponding filter has been applied. We can clearly see that the spatio-temporal filters reduce the number of result candidates and, thus, the number of refinements, dramatically. We can also observe that the probabilistic filter can reduce the number of refinements by 30% after applying the sequence of spatio-temporal filters. If we compare the overhead of the probabilistic filter (which is comparable to that of the spatio-temporal filter) with the additional cost that would be required to refine the candidates that are pruned by the probabilistic filter, we can observe that the cost required to perform the probabilistic filter can be neglected. This experiment shows that each of the filters incorporated in the UST-tree indeed pays off in terms of CPU cost.

Although I/O cost is not the bottleneck under our setting, an evaluation of the I/O cost is illustrated in Figure 17.6(c) for completeness. Filter cost here means all costs which occur during the tree traversal of the corresponding index structure, i.e. access to intermediate and leaf nodes. To derive the refinement I/O cost, we used the number of observation pairs which have to be refined, and assumed that each refinement yields one disk page access. Note, that the cost of a refinement can be much higher than one page access (e.g. if the Markov Chain, which can become very large does not fit in one disk page) under different settings. The filtering of the UST-Tree involves more page accesses since,



(a) $|\Lambda|$ for different values of τ (b) query extent (c) varying τ

(d) observation time interval (e) varying maximum speed (f) database size

(g) varying τ ($\forall$) (h) query extent($\forall$)

Figure 17.7: Experiments on Synthetic Data

the representation of the probabilistic diamonds requires more memory, which results in a larger tree. In contrast the R*-Tree based implementation requires less memory for the MBR approximations, but incurs more I/O cost due to larger number of refinements.

Since the above experiments imply that the most costly operation is the refinement of spatio-temporal diamonds, we will now take a closer look at the effectiveness of the three different methods on pruning spatio-temporal diamonds. The next experiments measure the number of spatio-temporal diamonds which have to be refined at the refinement step; these results can be directly translated to runtime differences of the different approaches. Note that a scan based approach would have to exhaustively refine all diamonds in the dataset.

Size of the Catalogue $|\Lambda|$

An important tuning parameter for the index is the size of the catalogue which is used for building the probabilistic diamond approximations. In Figure 17.7(a), it can be ob-

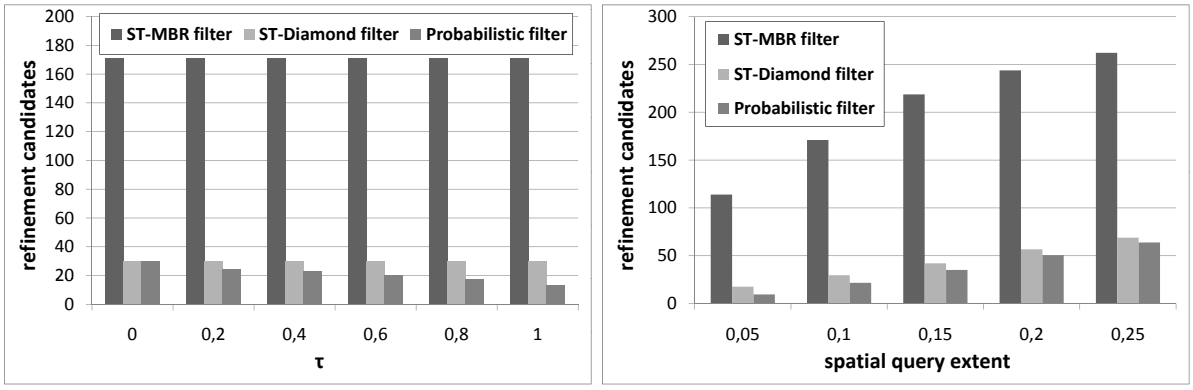
served that the filter effectiveness converges at around $|\Lambda| = 10$ (default value for the experiments). Depending on the query parameter τ, a too small catalogue yields up to twice as many candidates which have to be refined. Note that the number of refinement candidates does not decrease monotonically in $|\Lambda|$. In general a larger catalogue results in a linear function with a larger approximation error. However the step-function for the conservative approximation becomes smoother which results in a smaller approximation error. Because of these two contrary effects a larger catalogue does not always result in more filter effectiveness.

Query Parameters

The characteristics of the query have different implications on the index performance. Increasing the spatial extent of the query obviously yields more candidates since more diamonds in the database are affected (cf. Figure 17.7(b)). The spatio-temporal filter utilizing the diamond approximations becomes more effective in comparison to the ST-MBR-Filter . The percentage of the diamonds which can be pruned using the probabilistic filter remains rather constant (at around 30%) in comparison to the spatio-temporal filter. Another query parameter is the temporal extent of the query. Increasing the length of the query time window $T^{\square}$ increases the number of refinement candidates. The results are very similar to the results when increasing the spatial extent of the query; we do not include the comparison plot due to space limitations. Changing the value of τ obviously only affects the probabilistic filter (cf. Figure 17.7(c)). The higher τ is set, the more candidates can be pruned by the probabilistic filter. From a value of around 20% the candidates which have to be refined decrease linear in τ. The PST$\tau\forall$ query shows similar performance results (cf. Figures 17.7(g) and 17.7(h)) for the mentioned parameters. For the experiments we reduced the temporal query extent to 5, since the number of result objects for a PST$\tau\forall$ query is usually much lower than for a PST$\tau\exists$ query with the same query window.

Influencing Variables of ST Diamonds

The size of the spatio-temporal diamonds is generally affected by two parameters. One is the time interval between successive observations, since a larger interval results in more space which can be reached by the moving object between the two observations. For this experiment the number of time steps between successive observations in the data set was chosen randomly from the intervals on the x-axis in Figure 17.7(d). The second parameter is the speed of the object and has a similar effect. The speed is corresponding to the parameter ϵ, which reflects the maximum distance of points which can be reached by an object within one time step (cf Figures 17.7(e)). Since larger spatio-temporal diamonds usually result in more objects which intersect the query window, the number of refinement candidates increase when increasing these two parameters. Interestingly the effectivity of the ST-Diamond Filter decreases over the ST-MBR-Filter whereas the pruning effectiveness of the Probabilistic Filter increases. This shows, that the probabilistic filter copes better with more uncertainty in the data than the other two filters.



(a) varying τ (b) query extent

Figure 17.8: Experiments on Real Data ($\forall$)

Database Size

We evaluated the scalability of the UST-Tree by increasing the amount of observations. Figure 17.7(f) shows the results of these experiment. The number of results increase linearly with the database for a window query. The experiment also shows that the refinement candidates increase linearly for all filter steps.

Real Data

The experiments on the real world data, show similar behaviour as on the synthetic data. Thus we only show excerpts from the whole evaluation due to space limitations. Figure 17.8(a) illustrates the results for PST$\tau\forall$ queries when varying the value of τ. It is notable that the ST-Diamond Filter seems to even perform better (compared to the ST-MBR-Filter) on the real dataset. The reason for this is that the real dataset has much more inherent irregularity (regarding the locations and the movement of obejcts). This favors the ST-Diamond filter over the MBR approximation (since diamonds are more skewed as in Figure 15.1(b)). The probabilistic filter is apparently not affected by this situation. When varying the query extent (cf Figure 17.8(b)) the results resemble the results on the synthetic dataset.

17.3 Spatio-Temporal Nearest Neighbor Queries

For performance analysis, the sampling approach (Section 16) is divided into two phases. In the first phase the trajectory sampler (TS) is initialized (the adapted transition matrices are computed according to Algorithm 11). This phase can be performed once and used for all queries. In the second phase, the actual sampling (SA) of 10k trajectories (per object) is performed. The exact approach is denoted as EX. In our default setting during efficiency analysis we set the number of objects $|\mathcal{D}| = 10k$, the number of states $N = |\mathcal{S}| = 100$k,

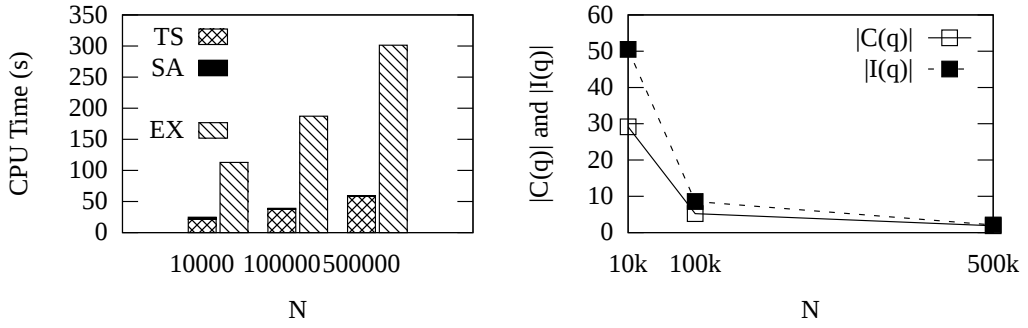


Figure 17.9: Varying the Number of States

average branching factor of the synthetic graph $b = 6$, probability threshold $\tau = 0$ and the length of the query interval $|T| = 10$. These parameters lead to a total of $110k$ observations (11 per object) and $100k$ diamonds for the UST-index.

Varying N.

In the first experiment (Figure 17.9) we investigate the effect of an increasing state space size N, while keeping a constant average branching factor of network nodes. This effect corresponds to expanding the underlying state space, e.g., from a single country to a whole continent. In Figure 17.9 (left) we can see that increasing N leads to a sublinear increase in the run-time of both the sampling approaches and the exact solution. This effect can be mostly explained by two aspects. First, the size of the a-priori model increases linearly with N, since the number of non-zero elements of the sparse matrix M increases linearly with N. This leads to an increase of the time complexity of matrix operations. At the same time, the number of candidates $|C(t)|$ and influence objects $I(t)$ decreases significantly as seen in Figure 17.9 (right) because the degree of intersection between objects decreases with a higher number of states, making pruning more effective. The actual sampling cost SA, which is too small to be noticeable in Figure 17.9 (left) decreases from 4s for 10k states to 0.7s for 500k states due to the smaller number of candidates and influence objects.

Varying b.

Figure 17.10 evaluates the branching factor b, i.e., the average degree of each network node. As expected, Figure 17.10 (left) shows that an increasing branching factor yields a higher run-time of all approaches due to a higher number of non-zero values in vectors and matrices, making computations more costly. Furthermore, in our setting, a larger branching factor also increases the number of influence objects, as shown in Figure 17.10 (right).

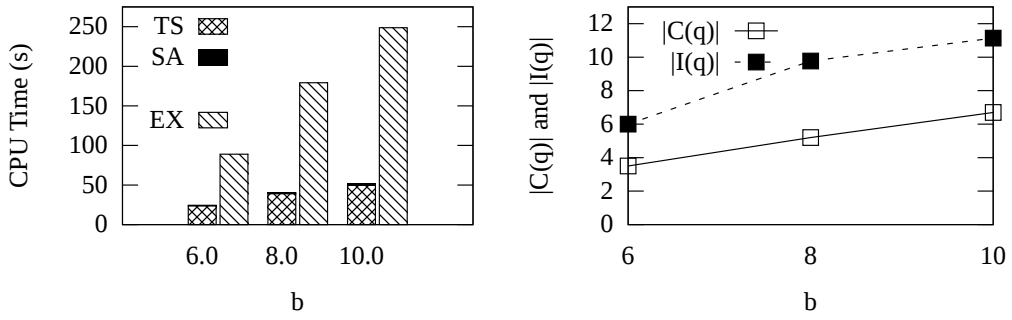


Figure 17.10: Varying the Branching Factor

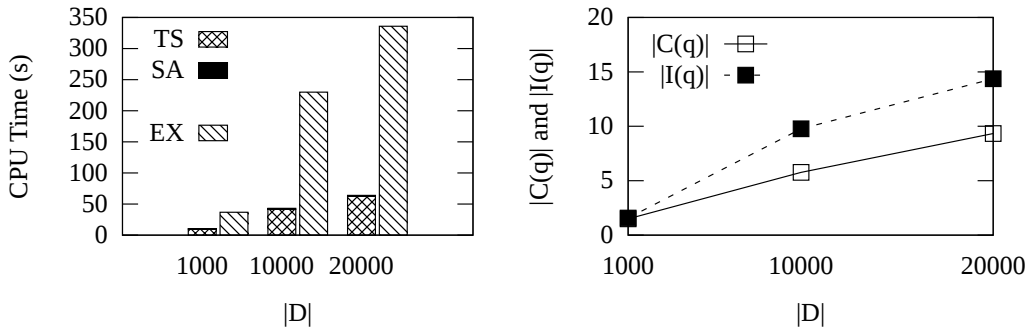


Figure 17.11: Varying the Number of Objects

Varying $|\mathcal{DB}|$.

The number of objects (Figure 17.11) leads to a decreasing performance as well. The more objects stored in a database with the same underlying motion model, the more candidates and influence objects are found during the filter step. This leads to an increasing number of probability calculations during refinement, and hence a higher query cost.

Varying s.

As a last experiment on P∀NN queries, we investigated the increase in runtime when varying the number of samples drawn. Increasing the number of samples (Figure 17.12) only affects the actual sampling process, but not the number of candidates or pruners. As expected an increasing number of samples makes the probability computation more expensive. In our example, the actual sampling time grows from 0.8s (1000 samples) to about 61s for 1Mio samples.

Real Dataset.

We conducted additional experiments to evaluate P∀NN queries on the taxi dataset (Figure 17.13). Since the underlying state space consisting of 3000 states is very small, we set $i = 5$

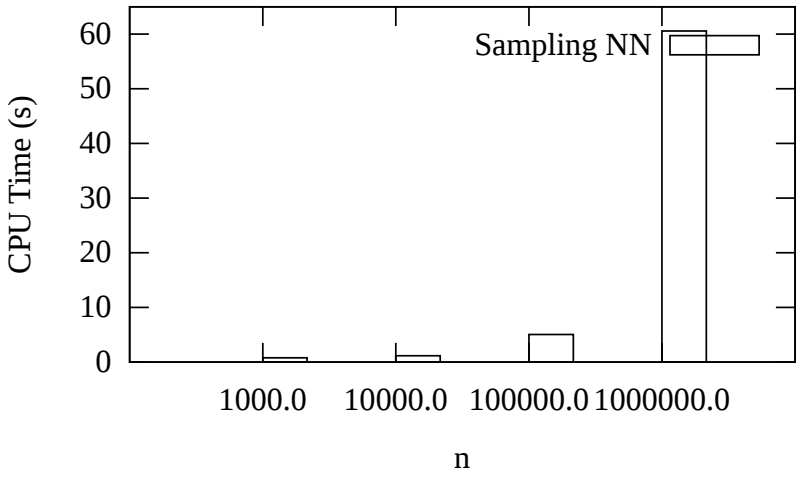


Figure 17.12: Varying the Number of Samples

and $v = 0.6$ in order to prevent uncertainty regions of objects to cover the whole network. Based on this dataset, we ran an experiment varying the number of objects between 1000 and 20000. The smaller size of the state space leads to a higher objects density, leading to a larger number of candidates and influence objects than the corresponding experiment on the artificial dataset. Additionally, the non-uniform distribution of taxis in the city is more dense close to the city center, making queries in this area more costly due to the higher number of candidates and pruners. Further note that in the real dataset, the motion patterns of objects are more diverse than on the synthetic data. There are taxis standing still, and taxis moving quite fast. Standing taxis have a larger area of uncertainty between observations, such that these objects reduce the performance of query evaluation.

17.3.1 Sampling Efficiency.

In the next experiment we evaluate the overhead of the traditional sampling approach (using the a-priori Markov model only) compared to the approach presented in Chapter 16 which uses the a-posteriori model again based on the artificial dataset. The first, traditional approach (TS1) discards any trajectory not visiting all observations. As discussed in Section 16.1, the expected number of attempts required to draw one sample that hits all observations increases exponentially in the number of observations. This increase is shown in Figure 17.14, where the expected number of samples is depicted with respect to the number of observations. This approach can be improved, by segment-wise sampling between observations (TS2). Once the first observation is hit, the corresponding trajectory is memorized, and further samples from the current observation are drawn until the next observation is hit. The number of trajectories required to be drawn in order to obtain one possible trajectory, i.e., the trajectory hits all observations, is linear to the number of observations when using this approach. We note in Figure 17.14, that in either approach at

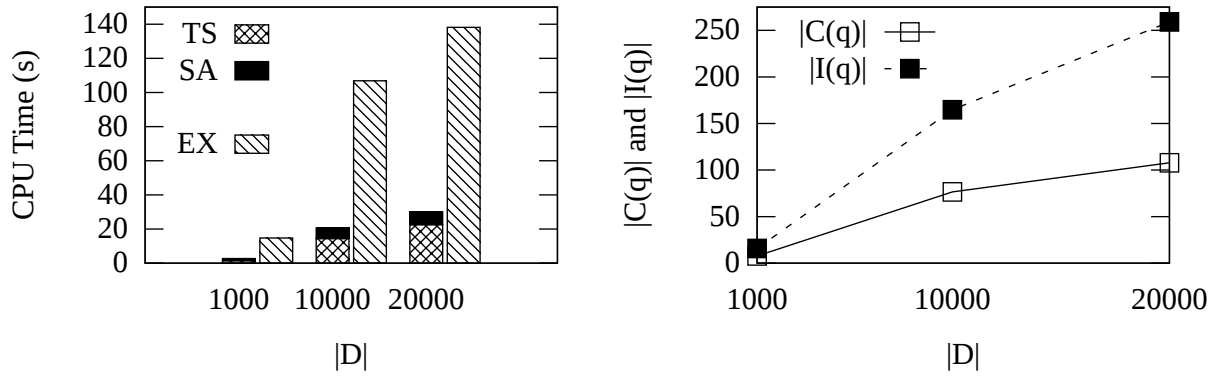


Figure 17.13: Realdata: Varying the number of objects

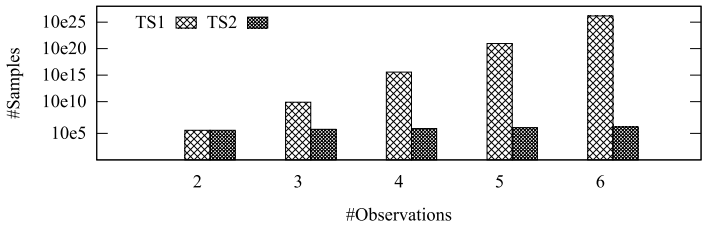


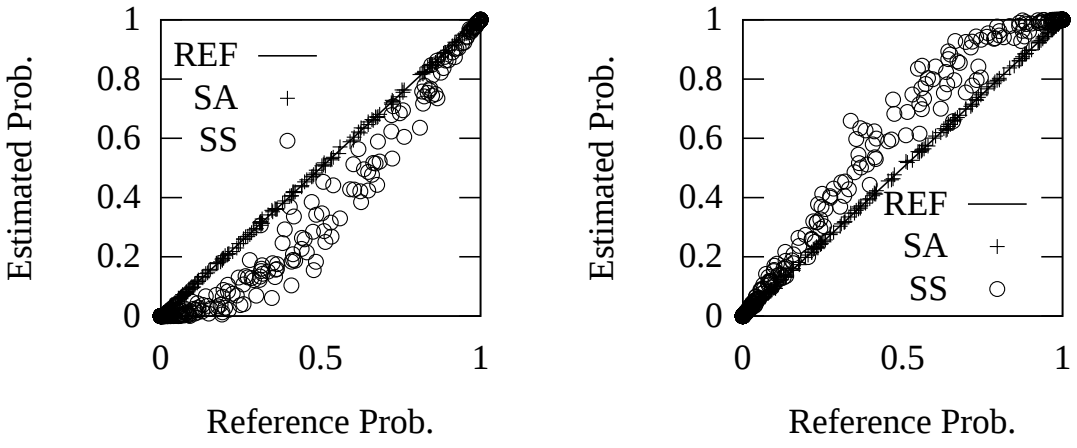
Figure 17.14: Efficiency of Sampling without Model Adaption.

least 100k samples are required even in the case of having only two observations. The reason is that by generation, trajectories follow a near-shortest path, which is a highly unlikely scenario using the a-priori Markov model. In contrast using the approach presented in Section 16, the number of trajectories that need to be sampled, in order to obtain a trajectory that hits all observations, is always exactly *one*.

17.3.2 Sampling Precision and Effectiveness.

Next, we evaluate the precision of our approximate P∀NNQ and P∃NNQ query and an aspect of a competitor approach [199]. The latter approach has been tailored for *reverse* NN queries, but can easily be adapted to NN query processing. Essentially, this approach performs a snapshot query $P\forall NNQ(q, \mathcal{DB}, \{t\}, \tau)$ for each $t \in T$. The probability $P\forall NN(o, q, \mathcal{DB}, T)$ is estimated by $\prod_{t \in T} P\forall NN(o, q, \mathcal{DB}, \{t\})$. $P\exists NN(o, q, \mathcal{DB}, T)$ can be approximated by $1 - \prod_{t \in T}(1 - P\exists NN(o, q, \mathcal{DB}, \{t\}))$. The scatterplots in Figure 17.15 illustrate a set of P∀NN and P∃NN probabilities on synthetic data ($v = 0.2$, $|T| = 5$). For each experiment, we estimate probabilities by our sampling approach (SA) (Section 16) with (10^4) samples and by the adapted approach of [199] (SS). We approximated the exact approach (REF) by drawing a very high (10^6) number of samples.

We model each case as a (x,y) point, where x models the reference (REF) and y the



(a) P∀NN (b) P∃NN

Figure 17.15: Effectiveness of Sampling, P∀NN and P∃NN

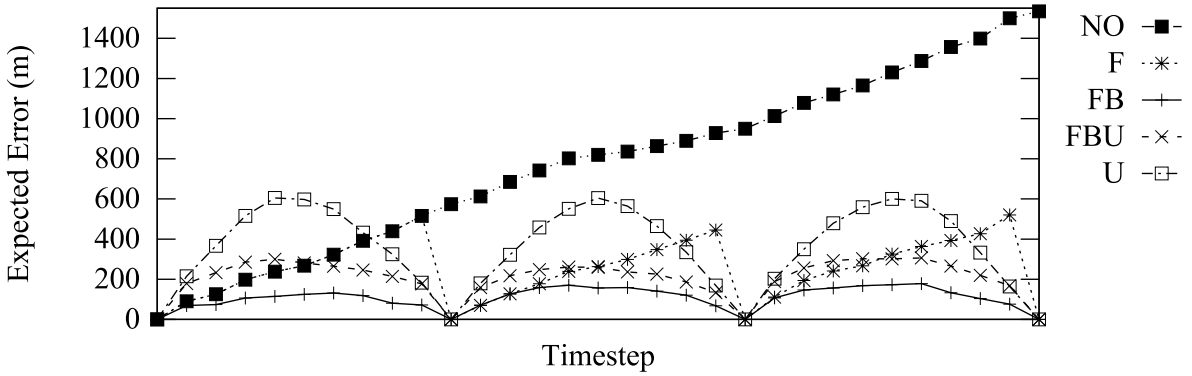


Figure 17.16: Realdata: Effectiveness of the Model Adaption

estimated probability (SA or SS). For (REF) the results always lie on the diagonal identity function depicted by a straight line. Probabilities of SA are very close to the diagonal, showing that our sampling solution tightly approximates the results of the exact P∀NNQ query. Using the snapshot approach, a strong bias towards underestimating probabilities can be observed for the P∀NNQ query. The snapshot-based P∃NNQ-query overestimates the results. These biases are a result of treating points of time mutually independent. In reality, the position at time t must be in vicinity of the position at time $t-1$, due to maximum speed constraints. This positive correlation in space directly leads to a nearest neighbor correlation: If o is close to q at time $t-1$, then o is likely close to q at time t. And clearly, if o is more likely to be close to q at time t, then o is more likely to be the NN of q at time t. This correlation is ignored by snapshot approaches. It can be seen that the systematic error of [199] is quite significant.

The number of samples required to obtain an accurate approximation of the probability of a binomial distributed random event such as the event that o is the NN of q for each time $t \in T$ has been studied extensively in statistics [90]. Thus the required number of samples is not explicitly evaluated here.

17.3.3 Effectiveness of the Forward-Backward Model.

In this section the effectiveness of the forward-backward model adaption in comparison to other approaches on the real dataset with a time interval between observations of 100 seconds. Figure 17.16 shows the mean error of these approaches, computed during each point of time, evaluated over a time interval of 30 tics (5 minutes). The mean error has been computed in leave-one-out manner, i.e. trajectories for computing the error have not been used to train the model in order to avoid overfitting. The figure visualizes the error of the a-priori model (NO) considering only the first observation, the model adapted by the forward phase only (F) and the forward-backward-adapted a-priori-model (FB) from this paper. We further implemented two additional approaches. The uniform approach (U), a competitor corresponding to [191, 188], discards all probability information of FB and, due to a lack of better knowledge, assumes all reachable states at a given time to have a uniform probability. The difference to the cylinders and beads approximation models presented in [191, 188] is that these models use conservative approximations that may include some (time, state) pairs actually having a zero probability for an object to be located at. Thus, our U approach is at least as good as the cylinders and beads approximation models in terms of effectiveness, regardless of the approximation type used. The approach FBU is equivalent to FB, however turning probabilities in the transition matrix are equally distributed instead of learning the exact transition probabilities from the underlying map data. First note that the approach not incorporating any observations (NO), yields significant errors compared to the remaining approaches. Clearly, observations can reduce errors and uncertainty during query evaluation. The forward-only approach (F) reduces this error, however the error is still high especially directly before an observation. This problem is solved by the forward-backward approach from this paper (FB). Note that even if the Markov chain is assumed to be uniformly distributed (FBU), the results are still good, but worse than with the actual learned probabilities (FB). This is good news, as it shows that even a non-optimally learned Markov chain can lead to useful results, however with a slightly higher error. This good performance comes from the fact that with a uniform transition distribution the diamond-shaped space of possible time-state pairs still has high probabilities in the center of the diamond, since trajectories near the center of the bead will have a higher likelihood than trajectories close to the beads boundary. This stands in contrast to the uniform approach (U) that models all states at the diamonds border to have the same probabilities as the states in the diamonds center; explaining why U performs worse than FBU. To conclude, combining observations with a sufficiently accurate transition matrix can produce the most accurate results.

17.3.4 Continuous Queries

In our experimental evaluation on continuous queries we compare the runtime cost and the size of the (unprocessed) result set for various sizes of the database and values of the threshold τ.

Increasing the number of objects stored in the database leads to an increase in the time

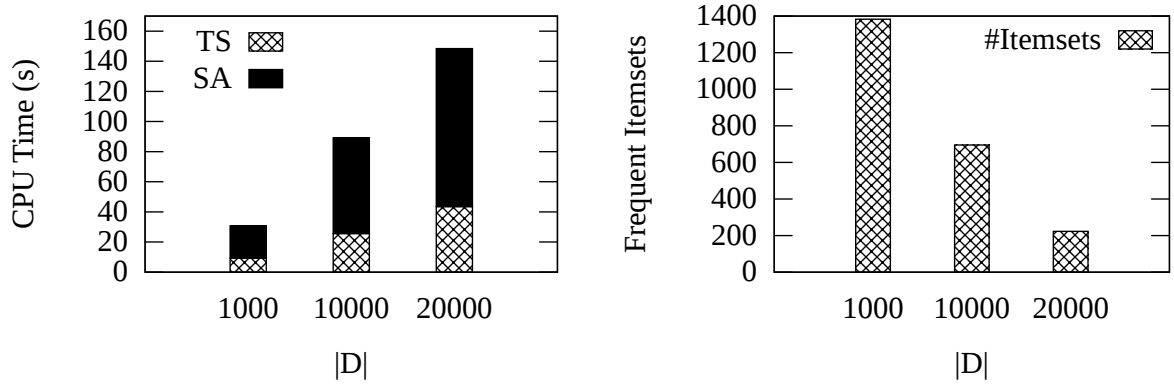


Figure 17.17: Continuous Queries: Varying the number of objects

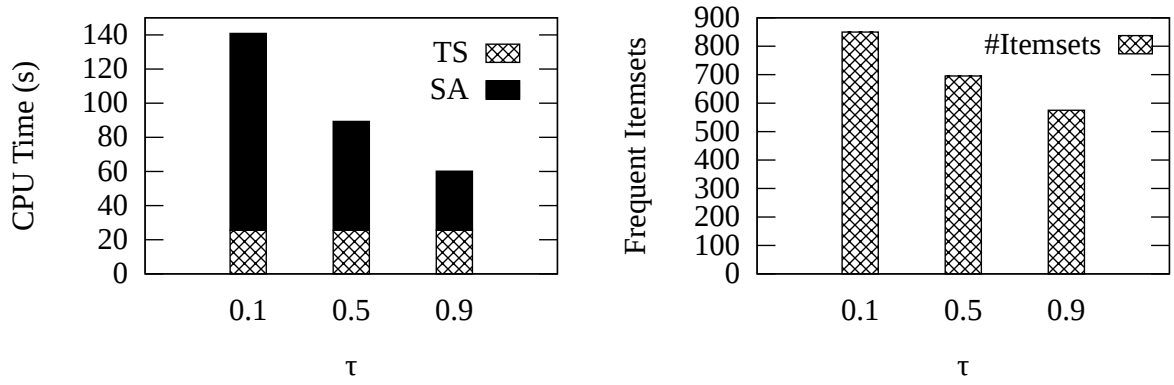


Figure 17.18: Continuous Queries: Varying τ

TS to compute the a-posteriori Markov model for each object (cf. Figure 17.17 (left)). This result is equivalent to the result for P∀NN queries, since a-posteriori models have to be computed for either query semantics. However, the time required to obtain a sufficient number of samples (SA) is much higher, since probabilities have to be estimated for a number of sets of time intervals, rather than for the single interval T. This increase in run-time is alleviated by the effect that the number of candidates obtained in the candidate generation step of our Apriori-like algorithm decreases (Figure 17.17 (right)). This effect follows from the fact that more objects lead to more pruners, leading to smaller probabilities of intervals, leading to fewer candidates.

The results of varying τ can be found in Figure 17.18. Clearly an increasing probability threshold decreases the average size of the result (Figure 17.18 (right)). Consequently, at the same time, the computational complexity of the query decreases as fewer candidates are generated. Figure 17.18(left) shows that the run-time of the sampling approach becomes

very large for low values of τ, since samples have to be evaluated for each generated candidate set. Similar to the Apriori-algorithm, the number of such candidates grows exponentially with T, if τ is small.

17.4 Summary

This experimental chapter has shown that the presented algorithms, both exact and approximative, allow to efficiently and effectively answer probabilistic queries on uncertain spatio-temporal data. The experimental results of this thesis should help to decide, for a given application, which algorithm to use. This decision must be based on a trade-off between result accuracy and run-time.

For systems that require precise result probabilities with a zero error tolerance, our theoretic results show, that only window queries can be answered efficiently on spatio-temporal data. If this type of query is sufficient for a given application, then exact algorithms can be utilized, as presented in Section 13.2. As we could see experimentally in Section 17, the run-time of this query can be significantly boosted by utilizing the proposed UST-index of Section15.

For applications requiring exacts answers to other types of queries not addressed in this thesis, the paradigm of equivalent worlds (see Chapter 3), helps to find an efficient solution, or help to find to quickly gain an intuition that no efficient solution may exist.

For applications that permit a small and bounded error, the Bayesian learning approach presented in Section 16 allows to approximate result probabilities in a very efficient way. This sampling based approximation can be used for any probabilistic query on uncertain spatio-temporal data, yielding a fall-back solution for any problem for which an efficient analytical solution may be out-of-reach.

Finally, the effectiveness results of Section 17 show that the a-posteriori Markov model which is build from the a-priori Markov model by learning the information provided by observation data, is able to tightly approximate the motion of moving objects. Surprisingly, this property even in the case where objects, as we expect them in most applications, do not have moving patterns following a Markov-chain, i.e., following a weighted random walk: We could see that trajectories following a near-shortest path could be precisely modelled. This strong property is the result of combining observation data from a spatio-temporal database with empiric transition information give by a Markov-chain. The observations are able to coarsely describe the true motion of objects, while the Markov chain is used only to predict deviation between discrete observations.

Chapter 18

Statistical Traffic Prediction in Road Networks

The Markov-Chain model suffers from the problem, that if a given mobile object o is at some network node n, then the probability distribution does not account for any past states visited. This problem results directly from the Markov assumption, that assumes that given the current position, future positions are conditionally independent of past positions (see Definition 56). In the previous chapter, this problem was solved by using observations in both future and past, to adapt the Markov chain dynamically. This way, the Markov chain at each point of time was adapted, to change the model from a simple weighted random walk model, to a model that ensures that any possible trajectory will conform to all observations, yielding high accuracy. In summary, this approach is very useful to interpolate the position between observations. Yet, dropping the Markov-assumption in order to consider an arbitrarily long history of an object, rather than its last state only, offers two very promising opportunities:

- The case of extrapolation, i.e., the case where we only have one observation, and try to predict the future without any future observation, the approach of Bayesean inference cannot be used, because there is no future information that can be used to adapt a-prioiri probabilities to a-posteriori probabilities. Without this adaption, we can only use the initial Markov-chain to model the future motion of an object. Such a model will equal a random walk, where an object may have a high probability of bouncing forth and back between two network nodes, without any meaningful destination. Such a model, clearly, does not project reality very well, since in practice, moving objects will try moving on a shortest path between their start and destination nodes, rather than randomly traversing the graph until they accidently reach their destination.

- a more precise model of the movement of an object between discrete observation may potentially improve the interpolation quality of all approaches presented in the previous chapters. In particular, in cases having a long time between discrete observation, a more sophisticated movement pattern model will be beneficial.

This chapter will focus on the first opportunity. Therefore, the Markov assumption will be relaxed, in order to consider the history of an objects movement in the prediction of its future position. While this can be achieved by simply using a higher order Markov-chain, such an approach will be highly limited due to the space and time cost exponential in the order of the Markov chain. The proposed solution will use a data structure, that allows to adaptively use a long history, i.e., a large list of previously visited nodes, in the cases where a long history does have a significant impact on the future motion. In cases where a long motion history does not have a significant impact on the future motion, only a sufficiently short history is used for prediction purpose. As experiments will show, this approach allows to efficiently predict the future motion of objects, while still allowing high efficiency for real time processing. Further experiments will show that in practise, only a rather small number of previously visited nodes is required to accurately predict the future motion, if we can assume that objects move on a shortest path.

18.1 Introduction

This chapter proposes a novel statistical approach to predict the density of any edge in a road network at some future point of time. The proposed method is based on short-time observations of the traffic history, i.e. the input for the traffic predictor are recent trajectories of the moving objects. The destinations and the following trajectories of the moving objects are unknown. Therefore, we need to estimate the future motion of the objects in the network. We assume that the moving objects will act rationally and choose the shortest path between their starting points to their destinations. Based on this assumption, we introduce a statistical approach for calculating the likelihood that a certain object is located at a certain network position at a certain point of time. Using the estimation for each object in the network, it is possible to estimate the traffic density at a certain position at a certain point of time. The allowed traffic capacity, i.e. the maximal traffic density that does not lead to a traffic jam, and the expected future traffic density of an edge at a certain point of time indicate the risk of a traffic jam. If the estimated traffic density exceeds the allowed traffic capacity of the edge, we can assume that a traffic jam is very likely to occur.

Formally this chapter offers a solution to the following problem: Given a set of moving objects in a road network, we want to estimate the traffic density (i.e. the expected number of objects located at a road segment at the same time) for all network segments at any future point of time. Besides the current position of each object, we additionally assume to have a short time history of each object, i.e. the recently visited network segments of each object in the network. Furthermore, it is assumed that the moving objects act rationally, e.g. each object moves along the shortest path between its starting point and its destination. This assumption can be used to estimate the destination of a moving object given only a prefix of its full trajectory.

The road network is represented as a directed weighted graph $G(V,E)$. V denotes the set of vertices that correspond to street crossings or points connecting two intersecting

road segments, and E denotes the set of directed edges that connect adjacent vertices and correspond to road segments. A weight is assigned to each edge that reflects the time any object requires to pass over the corresponding road segment. As an alternative, the distance of the way between the two adjacent vertices is used as weight and the time an object requires to traverse this segment is calculated by assuming some average speed on the road network. If the future trajectory of an object is known, its future location can be estimated by assuming an average speed at each road segment. Without this information, we would have to consider each possible location of the object for the future point of time. Since the number of possible future locations is often increasing strongly with the length of the time interval between now and the time of prediction, the prediction accuracy becomes rather small for prediction times that are not within the close future. However, assuming that each object moves along a shortest path and that the recently traversed trajectory is known for each object, it is possible to significantly restrict the potential destinations. Thus, our prediction model offers stable prediction results for a much longer period of time. The main contributions of this chapter are:

- A statistical traffic model that can be used to predict the traffic density in a network at any edge.
- A method to integrate the short time history of the network to significantly improve the prediction accuracy. Therefore, this method allows to make useful predictions over a significant period of time.
- Suitable algorithms and data structures to efficiently compute the prediction of the traffic density based on the provided information.

The remainder of this chapter is organized as follows. The next section contains a brief overview over existing approaches for traffic analysis and traffic jam prediction. In section 18.3, we introduce our statistical approach to estimate the traffic in a network at a certain position and at a certain point of time. An efficient method to speed up the prediction which is based on a suffix-tree is introduced in Section 18.4. In Section 18.5, we experimentally show the capability of our approach to make useful predictions about the traffic density. Furthermore, we illustrate the efficiency of our new algorithm when calculating these predictions and finally conclude the chapter with a short summary in section 18.6.

18.2 Related Work

In recent years, a lot of work has been published in the field of traffic data mining. One important problem in traffic mining is to detect areas with a significant high load of traffic. Some work has been published for the detection of traffic jams. Approaches for traffic jam detection are proposed in [76] and [119]. Both works address the problem of clustering trajectories, namely sets of short sequences of data like movements of objects. The resulting clusters indicate routes with a potentially high traffic load as the clusters represent

sets of objects that simultaneously move nearly the same route. While in [76] a model-based clustering algorithm is proposed that clusters trajectories as a whole, the approach proposed in [119] works on partitions of trajectories. Each trajectory is first partitioned into a set of line segments. Afterwards, similar line segments of different trajectories are grouped together in order to discover common sub-trajectories from a trajectory database. An important advantage of [119] against [76] is that it can also detect routes that do not necessarily span the complete trajectory of an object. Usually, the trajectory of objects moving in a traffic network are very long compared to the sections which form routes with high traffic density and thus, only reasonably small parts of a trajectory contribute to such routes.

Another approach for traffic jam detection is addressed by X. Li et al. in [128]. Their approach tries to discover *hot routes* in a road network. *Hot routes* are road segments that frequently or even regularly have a high traffic density and mostly lead to a traffic jam problem. The detection of *hot routes* is an important problem because each larger city has such *hot routes* that regularly block the traffic flow at rush hour and thus, traffic participants spend long times waiting in traffic jams. While the approaches proposed in [76] and [119] are individual traffic analysis methods because the traffic is computed by observing the motion of single individuals, the approach in [128] is based on the FlowScan algorithm which is also related to aggregate traffic analysis. It is able to extract *hot routes* by means of observing the traffic flow over some adjacent road segments. The algorithm does not completely fall into the category of aggregated traffic analysis because it considers more than the pure density of traffic on particular road segments. Therefore, the method can be considered to be a mixture between individual and aggregated traffic analysis.

Further approaches concerning traffic jam detection are based on the detection of dense areas of moving objects as proposed in [99]. This approach tries to find moving clusters in moving object databases. The difference of the addressed problem compared to clustering trajectories is that the identity of a moving cluster remains unchanged while its location and content may change over time. The same usually holds for traffic jams, in particular if the traffic jam is due to an obstacle that slows down the traffic. There are normally individuals that pass the obstacle at the beginning of the traffic jam, and thus, leave the traffic jam and those which arrive at the end of the traffic jam. Consequently, the contributors of the traffic jam change over time while the identity of the traffic jam remains.

A quite similar problem is addressed in [83] where areas of moving objects that remain dense in a long period of time are detected. This approach is quite related to our approach as it addresses predictions of dense traffics where the prediction concerns any time slot in the future. Furthermore, like in our approach the predictions are made on the basis of observations of the current motion. However, there is a big difference from our approach concerning the assumption of available information of the object motions. Previous traffic prediction and traffic detection methods assume that the traffic motion and thus, the object trajectories are known in advance. However, the future trajectory of an object is usually unknown in advance in our application scenarios.

Another challenging problem is the detection of general traffic patterns. There exist several approaches for traffic prediction by means of historical observations which are based

on regression analysis as proposed in [142]. Regression can be used to predict the future motion of individual objects as long as they do not move in a restricted environment as in our application. Another method concerning traffic prediction based on current traffic observations is the approach presented in [168]. In this work, the current traffic data is derived from a sensor network measuring traffic at certain locations in the traffic network. In the framework proposed in [168], the sensor network includes about nine hundred measurement stations. The data is collected in a data warehouse and used to infer interesting patterns. This kind of system may be used to learn patterns on the observed data which could be used to predict traffic jams. This method falls into the category *aggregate analysis* and mainly differs from our approach as it aggregates the traffic at certain road segments instead of observing single individuals.

A further related topic in traffic mining is the detection of suitable traveling paths for individuals that want to travel to a certain destination in a possibly most cost-efficient way. There is a lot of published work related to fastest path computations [58, 60, 73, 143, 97, 98, 100, 162]. However none of these proposals takes the actual traffic into account. An efficient technique for fastest path computation taking traffic patterns into account has been addressed by H. Gonzales et al. [78]. The authors propose an adaptive navigation method based on historical traffic patterns mined from a large set of given traffic data.

Another field related to traffic mining is graph mining which has attracted a lot of attention in recent years. At first sight, graph mining seems to be closely related to traffic mining as traffic flows normally occur in a network graph, e.g. a road network or the internet. However, most graph mining approaches deal with the topological structure within graphs or subgraphs. A lot of graph mining approaches aim at finding interesting patterns within graphs. A comprehensive survey of graph mining techniques is given in [42]. Although the approach also employs a static network graph topology, the method proposed in [186] discovering *center-piece subgraphs* is related to our approach. The *center-piece subgraph* problem is to find the node(s) and the resulting subgraph, that have strong connections to all or most of a given set of query nodes. Usual applications are connectivity mining in social networks, gene regularity networks and viral marketing. In a traffic network, we usually have certain places of preferred travel destinations or starting points. Such kind of *hot spots* can be malls, theme parks, city centers, commercial centers, conjunction to highways and so on. These kind of hot spots can be used as query points in order to find *center-piece subgraphs* which indicate places of expected high traffic. Similar measures like "Closeness Centrality" and "Betweenness Centrality" [138] which are traditionally used for mining in biological interaction networks can be applied to identify road segments with high risk of traffic jams.

18.3 Statistical Traffic Model

This section will formalize our view on road networks and the traffic that can be observed on them. Furthermore, we will discuss a statistical model that allows to predict future states of the network under the knowledge of the current state and a short time history.

18.3.1 Traffic Density in a Network

A traffic network is modeled as a graph $G(V, E)$ where the vertices represent destinations and crossings. The edges represent ways or streets between the vertices. A walk is a sequence of edges $w = (e_1, .., e_n)$ where successive edges are connected, i.e. $e_i = (v_l, v_k) \Rightarrow e_{i+1} = (v_k, v_m)$ with $v_l, v_k, v_m \in V$. An object o may travel on this network from one vertex v_1 to another one v_2 by following some walk $w = (e_1, .., e_n), e_i \in E$ where e_1 is starting with v_1 and e_n is ending with v_2.

The point of time t_i where the object reaches v_2 depends on the speed the object is traveling at on each edge. Therefore, we assume that there is a maximum speed for objects traveling on a certain edge e_i $speed_{e_i}$. Knowing the length of edge e_i $length(e_i)$, we can determine the time it takes object o to travel from v_l to v_k via e_i. Thus, given the walk $w = (e_1, ...e_n)$ we can calculate the time it takes object o to follow w by

$$time(o, w) = \sum_{i=1}^{n} \frac{length(e_i)}{speed(e_i)}$$

To find out the position of object o at time t traveling on a walk $w = (e_1, .., e_n)$, we have to calculate the $time(o, (e_1, .., e_i))$ for $i = 2$ to $i = n$. We can stop at the first edge for which $time(o, (e_1, .., e_i))$ is larger than t, because o will not reach e_i in a time shorter than t. As a result, we know that o will travel on edge e_{i-1} at time t.

After describing the movement of individual objects in the network, we will turn to describing the complete state of the network. Thus, we define the density on edge e_i at time t as the number of objects traveling on e_i at time t. Formally, the density is defined as:

Definition 67 (Traffic Density)**.** *Let $G(V, E)$ be a traffic network and let $O = o_1, .., o_m$ be a set of objects traveling on the network. Furthermore, let $\rho : E \times O \to \{0, 1\}$ be the following indicator function*

$$\rho(o, e) = \begin{cases} 1 & \text{if } o \text{ is on } e \\ 0 & \text{else} \end{cases}$$

Then, the traffic density on edge e is defined as:

$$density(e) = \sum_{i=1}^{m} \rho(o_i, e)$$

Clearly, it is possible to determine the density of each edge at the current time t when observing the network.

Additionally, it is possible to compute the density for any future point of time $t + \Delta t$ if all objects $O = \{o_1, .., o_m\}$ and their corresponding paths w_{o_i} are already known at time t. As mentioned above the position of object o_i at the point of $t + \Delta t$ can be derived easily when knowing the walk o_i is traveling on.

The problem of traffic prediction is caused by the absence of knowledge about the walk w an object o is traveling on. In other words, we can observe each object on the network at time t but without knowing its route, we cannot exactly tell its position at time $t + \Delta t$.

Though we cannot tell the exact density of each edge at some future time $t + \Delta t$, it is possible to calculate an expected density employing the available knowledge about the objects and their behavior. Generally, our model for determining the expected density is based on the probability that a given object o is traveling on edge e at time $t + \Delta t$, $Pr[o, e, t + \Delta t]$.

This probability depends first of all on the existence of a walk that allows o to travel on edge e at the time of prediction $t + \Delta t$. If there is no walk allowing o to reach e in Δt time, then $Pr[o, e, t + \Delta t]$ can be considered to be zero.

After finding all walks $W = \{w_1, .., w_l\}$that allow o to be at e at time $t + \Delta t$, we can sum up the likelihoods $Pr[o, e, w_i]$ that o would take walk w_i:

$$Pr[o, e, t + \Delta t] = \sum_{i=1}^{l} Pr[o, e, w_i]$$

To determine $Pr[o, e, w_i]$, we assume that w_i is the result of a Markov chain on the network where the vertices correspond to the states and the edges to the allowed transitions. The chain is started at the current position of o. Each time o reaches a new vertex v, o has to decide for one of $deg(v) + 1$ options. $deg(v)$ denotes the degree of v, i.e. the number of adjacent edges. Thus, an object can either stop traveling at the vertex v or take any of the adjacent edges to continue its journey. To find out the likelihood that o follows the walk w, we need to assume a probability distribution describing the likelihood of each of these options. Formally, we can calculate the probability that object o follows walk $w = (e_1, .., e_n)$ where $start_i$ and end_i denote the starting and ending node of edge e_i as:

$$Pr[o, e_n] = \prod_{i=1}^{n} Pr_o[end_i | start_i]$$

$Pr_o[end_i | start_i]$ is the likelihood that o enters e_i under the condition of being previously on node $start_i$. Let us note that we do not distinguish whether o intends to stop at the end_i or continues its travel. In a classical Markov chain $Pr_o[end_i | start_i]$ usually depends on the previously visited edge e_{i-1}. However, in order to keep our framework as general as possible, we do not limit our method to a certain type of distribution and thus, allow arbitrary probability distributions. For example, we might assume that the underlying probability distributions are uniform. In this case, the likelihood that o is taking walk w follows the random walk assumption, i.e. at each node an object would take any of the given options with the same likelihood with no regard of any global destination. However, since objects in a traffic network usually behave more rationally, we will introduce more sophisticated probability distributions in the next subsection.

After describing the likelihood $Pr[o, e, t + \Delta t]$ that object o will be at edge e in Δt time, we can now calculate the expected density for edge e at $t + \Delta t$.

Definition 68 (Expected Density)**.** *Let $G(V, E)$ be a traffic network and let $O = \{o_1, .., o_m\}$ be a set of objects traveling on the network. Then, the traffic density on edge e at time $t + \Delta t$ is defined as:*

$$density(e, t + \Delta t) = \sum_{i=1}^{m} Pr[o_i, e, t + \Delta t]$$

18.3.2 The Shortest Path Assumption

Though the expected density allows us to predict the expected state of a traffic network for any future point of time, its applications pose serious problems. First of all, the prediction is strongly dependent on the underlying probability distributions. Thus, if these distributions do not model the behavior of the objects well enough, the expected density will significantly differ from the real density after a short period of time. A second problem is the computational cost of determining all walks between the current position of an object o and a future position e. The number of possibilities we have to check is exponentially increasing with Δt. Thus, finding all walks allowing o to travel on edge e at $t + \Delta t$ is very expensive for larger values of Δt.

Fortunately, the random walk assumption made above is not realistic for most traffic networks and we can employ more realistic assumptions to derive more suitable probability distributions and reduce the number of walks.

For example, a driver traveling from New York to Los Angeles would not randomly decide at each highway intersection in which direction he drives next. The reason for the more rationale behavior in traffic networks is that each object has usually a predefined destination, it wants to reach as fast as possible. Furthermore, the topology of the network is known to each object and thus, the object does not have to stray through the network until it accidentally reaches its destination. Since each object wants to reach its destination as fast as possible, we can assume that each object travels along a shortest path where each edge e is weighted by the time it takes to traverse it, i.e. $\frac{lenght(e)}{speed(e)}$. A path in contrast to a walk is not allowed to contain the same vertex twice. We will refer to this observation as the *Shortest Path Assumption*.

Though the general framework for computing the expected density can remain unchanged, the shortest path assumption has a major impact on the quality of prediction and the computational complexity.

A first implication is that in order to determine whether object o might be at edge e after the time period Δt, we only have to consider the shortest paths of the current position of o to the end vertex of e. If there is no path ending with edge e, then o travels on edge e with a probability of 0%. If e is the last edge of some shortest path, we can calculate the time period o would travel on e. Only if $t + \Delta t$ is within this time period, it is possible to observe o on edge e at the time of prediction $t + \Delta t$. Let us note that it is not necessary to consider any other shortest path because an object traveling on any other shortest path must arrive at the end of e at the same time. To conclude the shortest path assumption significantly reduces the number of walks that have to be considered.

A further implication of the shortest path assumption is that it is possible to find meaningful probability distributions that can be used to determine the likelihood that object o travels along path p.

We know that each object o heads towards one predefined destination v_o^{dest}. Furthermore, we know that o travels along a shortest path to reach v_o^{dest}. Thus, the set of all possible paths o could follow, is the union of all shortest paths to any possible destination. Now the likelihood that o travels along path $p = (v_1, .., v_n)$ depends on the probability that v_n is o's target, $Pr_{dest}[o, v_n]$. Without further knowledge we might assume that each destination is equally likely. Additionally, it is possible to increase the likelihood of more popular destinations to integrate domain knowledge. Let us note that in the case that there is more than one shortest path leading to v_n, we assume that all paths are equally likely.

After assuming a distribution over all destinations, it is possible to derive local probability distributions that can be employed to estimate the likelihood that object o travels on a certain edge e. Therefore, we need to sum up the probabilities for each shortest path containing edge e. Formally, we can define this probability as follows:

Definition 69 (Visiting Probability)**.** *Let $G(V, E)$ be a traffic network, let o be an object having the current position v_{start} and, let $sp(v_{start}, v)$ denote the set of shortest paths from v_{start} to any other vertex $v \in V$. Furthermore, let $\hat{P}_{v_{start}}(e_n) = \{p | p \in sp(v_{start}, v) \wedge v \in V \wedge e_n \in p\}$ be the set of all shortest paths beginning with v_{start} and containing the edge e_n. Now, the probability that o follows the path $p = (v_{start}, ..., v_k)$ $Pr_o[p]$ is defined as :*

$$Pr_o[p] = \frac{1}{|sp(v_{start}, v_k)|} \cdot Pr_{dest}[o, v_k]$$

where $Pr_{dest}[o, v_k]$ is the likelihood that v_k is the destination of object o. Then, the probability under the shortest path assumption that object o travels on edge e_n is defined as follows:

$$Pr_{sp}[o, e_n] = \sum_{p_i \in \hat{P}(v_{start}, e_n)} Pr_o[p_i]$$

The probability $Pr_{dest}[o, v]$ describing the likelihood of each possible destination has an important impact on the accuracy of the prediction. Furthermore, under the shortest path assumption this likelihood depends on the path $p^o_{history}$,i.e. the path o has already traversed until the current point of time. If $p^o_{history}$ is unknown, we generally have to assume that all vertices are possible destinations of o. However, knowing $p^o_{history}$ allows us to prune some of these destinations. Since o is traveling on a shortest path, we can exclude all destinations for which there exists no shortest path starting with $p^o_{history}$. Thus, knowing the previous movement of each object o significantly reduces the number of possible destinations and thus, allows us to find a better estimation of $Pr_{dest}[o, v]$.

To conclude, the shortest path assumption can be derived from assuming that all objects have knowledge of the network topology and try to reach a certain destination as fast as possible. Based on the shortest path assumption, we can derive more reasonable probability

distributions for the decisions each object makes at some vertex. Thus, it is possible to find a more suitable expected density for the edges in the traffic network.

18.4 Efficient Traffic Prediction

In the previous chapter, we defined the expected density for single edges in a traffic network at a certain time of prediction. In this chapter, we will turn to calculating the complete density in a network at some future point of time consisting of the expected densities of each edge in the network. After introducing a straight-forward method to calculate this expected network density, we will introduce a data structure allowing a much more efficient computation of density predictions.

18.4.1 Traffic Density Prediction

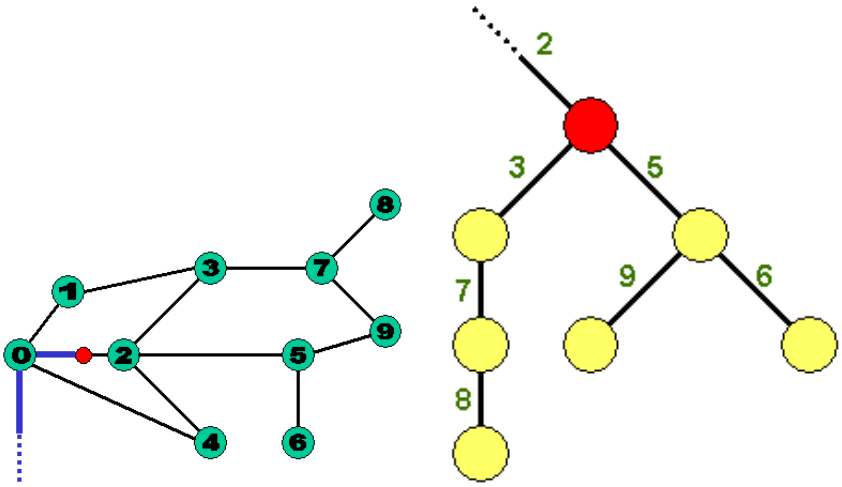
The goal of our approach to is to predict the state of a traffic network for a future point in time or even a time period in the future. Therefore, we first of all formalize the expected density in a traffic network.

Definition 70 (Expected Network Density)**.** *Let $G(V, E)$ be a traffic network and let $O = \{o_1, .., o_m\}$ be a set of objects traveling on G under the shortest path assumption. For each object $o_i \in O$, we know a short time history $p^o_{history}$ containing the path o_i has traversed before the current time t. Furthermore, the destination of each object o_i is unknown. Then, the **Expected Network Density** at time $t + \Delta t$ is defined as the set of expected densities for each edge e: $density(e, t + \Delta t)$.*

The Expected Network Density consists of the complete traffic density that can be expected to be observed at some future point of time.

In the following, we will discuss a straight-forward method for calculating the expected network density, i.e. the expected density of each edge in the network at the time of prediction $t + \Delta t$.

The basic idea of the following method is to determine all possible positions for each object o at prediction time $t + \Delta t$. Thus, we increase the density of each edge e by the probability $Pr_{sp}[o, e]$ if o might visit e at the time of prediction. To find out all possible positions and their corresponding likelihoods, we first of all have to determine all possible destinations. As mentioned before, the number of possible destinations depends on the path $p^o_{history}$ that o has already traversed. Therefore, we start with the first known position of o, i.e. the first node in $p^o_{history}$, and employ Dijkstra's algorithm to determine all shortest paths to any other node in the network. Now, to determine all possible positions of o at time $t + \Delta t$, we only have to consider the shortest paths being extensions of $p^o_{history}$. Each of these extending paths leads to a still possible destination. Thus, we follow each of the paths p for the time period Δt and thus, determine a possible position of object o. Now, the expected density of the edge corresponding to this position is increased by $Pr_{sp}[o, p]$,



(a) Network graph (b) Suffix tree

Figure 18.1: Example of a network graph and the corresponding suffix tree used to efficiently compute an objects probability distribution.

i.e. the likelihood that o travels along path p. After processing each possible position for each object in the system, the expected network density is derived.

A variation of this method can be applied if we are not only interested in the traffic density at a special point of time $t + \Delta t$, but in the expected density at all points of time between t and $t + \Delta t$. In this case, the prediction of the expected density is represented by a time series displaying the expected change of traffic on a given edge. However, computing the time series is quite similar to computing a single prediction. For each path p, extending $p^o_{history}$, we traverse p and update the edges of p for the period of time o might travel on them. Whenever o could enter a new edge the expected density is increased by $Pr_{sp}[o, p]$. Correspondingly, the expected density has to be reduced each time o leaves some edge e.

Though this method can be employed to determine predictions according to the traffic model introduced in section 18.3, it has serious short-comings from a computational point of view. The problem is that in order to determine the possible paths of object o, it is always necessary to consider each node of the network and determine all possible shortest paths starting with its first known position. This poses an enormous computational overhead because some of the paths are computed for multiple objects. However, since the usefulness of a prediction is rather perishable, a fast computation of the prediction is mandatory. Thus, in order to derive predictions in efficient time, a solution has to be found avoiding this computational overhead at prediction time and allowing efficient density predictions for the complete network.

18.4.2 A Shortest Path Suffix Tree

In the following, we will present a data structure that is significantly speeding up the computation of the expected network traffic density. The core idea is to store all possible shortest paths in a compact data structure. Thus, the computation of shortest paths at prediction time can be avoided.

Assuming that there exists an unique identifier denoting each node in the network, we can use these node identifiers as alphabet and represent each path as a string over this alphabet. Our algorithm needs an efficient way to determine all shortest paths being extensions of the already observed history of a given object o. Considering each shortest path as string, we need to find a way to efficiently determine all suffixes extending the prefix represented by $p^o_{history}$. Therefore, we propose to store all shortest paths that can be found in the given network in a suffix tree.

The suffix tree is well known in text processing and bio informatics for its space-efficient storage of massive amounts of string data. Formally, a suffix tree ST for string $S = S[0..n-1]$ of length n over the alphabet A is a tree with the following properties:

- ST has exactly $n+1$ leaf nodes, numbered consecutively from 0 to n.
- All internal nodes (except the root) have at least two children.
- Edges spell non-empty strings.
- All edges from the same node start with a different element of A.
- For each leaf node i, the concatenation of all edges from the root node to i matches $S[i..n-1]$.

In order to employ the suffix tree for our problem, we store all shortest paths in the given network in the suffix tree. Therefore we use the all-pair-shortest-path-algorithm by Floyd and Warshall to efficiently derive all possible shortest paths. Afterwards, all shortest paths are converted to strings over the alphabet of node identifiers and stored in the suffix tree. In this suffix tree, each direct son of the root represents a vertex v in the network and the corresponding sub tree represents all shortest paths starting with v. Let us note that each path in this sub tree corresponds to a shortest path and the paths to the leaf node represent shortest paths that are maximal, i.e. it is not possible to extend these paths to any longer shortest path. Each inner node v_n of the suffix tree represents a crossing in the network where some object o could arrive after traversing the path corresponding to its history $p^o_{history}$. The sons of v_n represent all possible shortest paths extending $p^o_{history}$. Figure 18.1(a) illustrates an example of an object traversing a network graph. The corresponding suffix tree representing all possible destinations is depicted in Figure 18.1(b).

To efficiently calculate the expected network density, it is not sufficient to directly access the information about the existence of a shortest path. Additionally, the likelihood that an object o follows some path p is of great importance. Therefore, we additionally store the probability distributions describing $Pr_o[end_i|start_i]$ in the tree, i.e. the likelihood

o would turn into the direction of the node end_i after reaching $start_i$. In our model, this probability depends on the cumulated likelihood that o takes any of paths being extensions of the edge $(start_i, end_i)$. In the tree, these paths are represented by the sub tree under the node end_i. To speed up the computation of the likelihood of each path during prediction, we add up the likelihoods of possible directions right after generating the tree. Therefore, we first of all mark each ending point of each path, with the likelihood that o would take this path. Let us note that inner nodes are valid ending points as well. Afterwards, we assign the cumulated likelihood over all paths extending edge $\hat{e}$ to $\hat{e}$ in the tree. Let us note that for any node e in the network there usually exist multiple edges $\hat{e}$ in tree, one for each possible prefix. Due to this modification, it is now possible to calculate the likelihood that some object o might visit edge e while traversing the tree.

To calculate the expected network traffic density using the proposed shortest path suffix tree, we can proceed as follows. For each object o, we enter the tree traversing along the string corresponding to the already observed path $p^o_{history}$. After reaching the node in the tree corresponding to the current position of o, we can derive all possible positions of o at $t + \Delta t$. Therefore, we traverse every path in the sub tree under the current position of o and calculate the likelihood that o would travel this path during traversal. Traversing each path is stopped if extending the path would demand more time than Δt. Finally, we add the current likelihood to the expected density at the edge corresponding to the current position of o and continue by extending the next path in the sub tree.

To conclude, employing a shortest path suffix tree allows us to avoid shortest path computations during traffic density prediction. Furthermore, the number of edges that have to be traversed for prediction is also reduced to necessary sub paths.

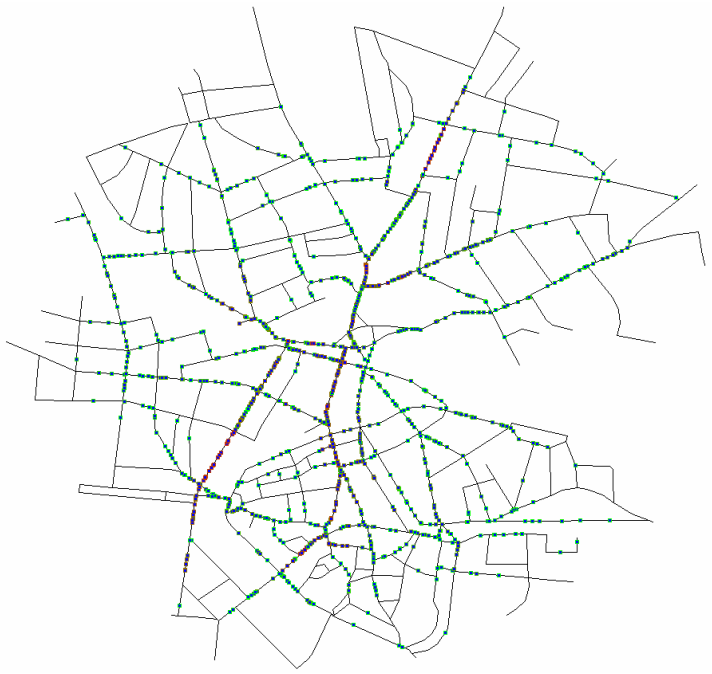


Figure 18.2: Traffic network graph with simulated cars used as experimental test bed.

18.5 Experimental Evaluation

In this section we show the capability of our approach to make useful predictions about the traffic density and illustrate the efficiency of our new algorithm when calculating these predictions. For all experiments, we simulated the traffic in a realistic traffic network as depicted in Figure 18.2 containing about 679 road segments and 533 intersection nodes. Our traffic simulator contains about 1250 cars (illustrated by small dots in Figure 18.2) moving from individual starting points to their destinations on a shortest path. The starting points as well as the destinations are equally distributed over the entire network graph. Here, each car moves with a certain velocity which is assumed to be constant during the whole journey. The velocities of the moving cars are randomly selected for each car and it took about 60 minutes until all cars have reached their destinations. If not stated otherwise, as soon as a car has reached its destination it was removed from the network and, thus, did not contribute to the traffic anymore.

All experiments are based on java implementations. The runtime experiments were conducted on a dual core Opteron Dual Core processor with a clock time of 2.6 GHz and 32GB of RAM.

18.5.1 Experiments on Quality of the Traffic Density Prediction

The first experiments concerns the quality of our traffic density predictions. The traffic density of a road segment is simply given by the number of cars that pass through this

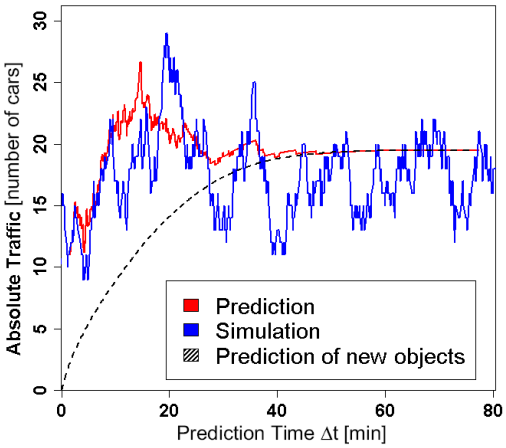
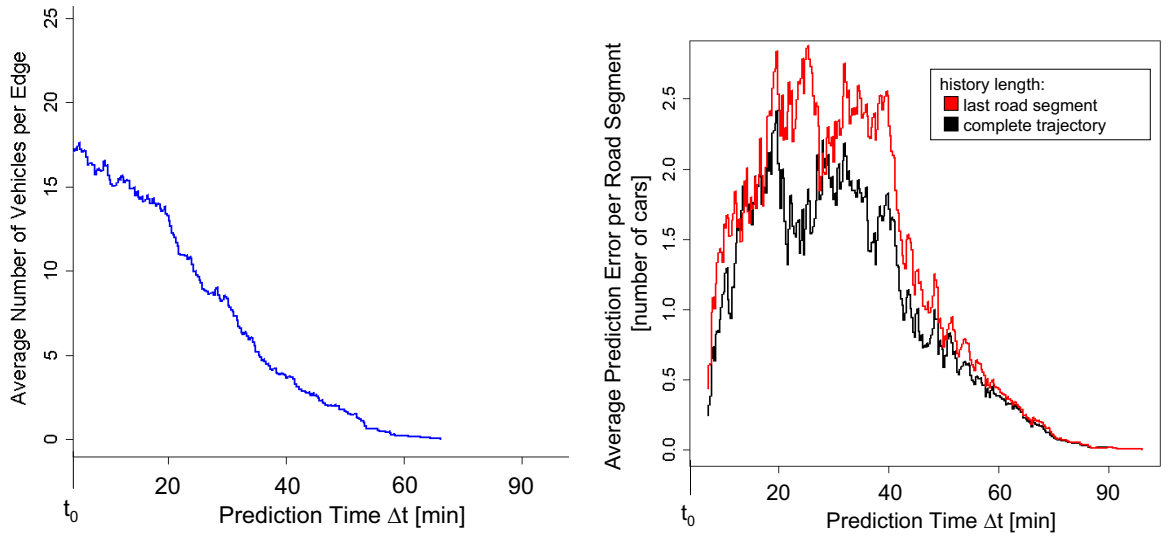


Figure 18.3: Prediction using a spatial temporal poisson model for the entry of new cars.

road segment at a certain point of time. In order to show the quality of the traffic density prediction, we continuously measured the traffic density *prediction error* during a certain range of time. The *prediction error* is computed by the difference between the predicted traffic density and the observed traffic density for a road segment. In our experiments, we use the parameter *prediction time* Δt which denotes the the forecasting horizon. In other words, the prediction time denotes the difference between the time the actual traffic is measured, (i.e., the time the traffic prediction is related to) and the time at which the traffic prediction was done.

Generally, in our experiments we only consider those cars that are existent in the road network at prediction time, i.e. all objects that enter the network graph after the time the prediction was done are not considered. However, in realistic scenarios new cars continuously enter the network. In order to evaluate traffic predictions under these circumstances, an additional statistical model would be required. For example, the entry of new objects in the network can be modeled using a spatial temporal Poisson-process. The prediction based on such a model is depicted in Figure 18.3. Obviously, the absolute prediction error increases with the number of new objects entering after the time the prediction was done. The rationale of this is that we have no information of the new cars while the number of cars which are considered for the prediction diminishes. The expected number of cars which are considered for the traffic prediction is the difference between the predicted number of cars and the expected number of new cars. This number approaches zero when all objects considered for the traffic prediction have reached their destination. In the following experiments we only focus on objects which are present in the network at the time the prediction is done and do not allow objects to enter after that.

Since the prediction error is an absolute value measured in number of cars, the quality of the prediction depends on both the prediction error and the number of cars on the corresponding road segment. If not stated otherwise, we averaged the prediction error over a set of road segments. In order to achieve more representative results, we measured



(a) Number of cars.

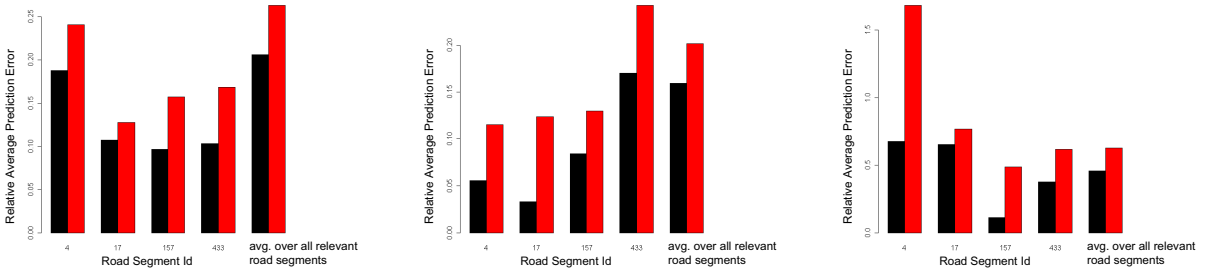
(b) Average prediction error

Figure 18.4: Impact of the Markov assumption.

the prediction quality only for a subset of road segments. Here we left out those road segments that contain only very low traffic over the measured time. Thereby, we try to avoid that the quality results are inherently biased by road segments with low traffic which are expected to yield high prediction quality. Since there are a lot of such kind of road segments with little traffic in our traffic network we did not consider them in order to obtain fair quality measurements. In the remainder, we will call the set of road segments taken into account for the quality measurements *relevant road segments*. This set contains twenty road segments.

The average number of cars on a relevant road segment is given in Figure 18.4(a). It shows that the number of cars decreases with the running time of the simulation as the cars which reach their destination are removed from the traffic simulation. Although the overall number of objects in the simulation in fact decreases monotonically, here we do not observe a monotone decrease in the number of cars because we only counted the objects on the relevant road segments which can fluctuate a little bit.

Figure 18.4(b) shows the average prediction error w.r.t. the prediction time Δt (i.e. the forecasting horizon). The figure presents two curves, one curve depicts the prediction error when considering the complete history of each car for the prediction. The other curve represents the prediction error when taking only the last two passed road segments into account. Both curves have similar characteristics, because within the prediction of the close future the error increases drastically with increasing prediction time. This is due to the fact that the number of possible locations for each car is initially very small and increases drastically when the car passes through the first crossings. But with ongoing prediction time, the absolute prediction error decreases again. The rationale of this effect is the decreasing number of cars in the simulation. Less cars in the simulation obviously lead



(a) prediction time = $0 - \infty$ (b) prediction time = 18-22 min (c) prediction time = 38-42 min

Figure 18.5: Relative prediction error over certain intervals of prediction time.

to a smaller absolute prediction error. But it can clearly be observed that the prediction based on the complete history has a significantly better quality than the prediction based on only the last two road segments. Generally speaking, both predictions have a good prediction quality, at least in the first few minutes. To quantify the prediction quality, we measure the *relative prediction error* defined as

$$\text{relative prediction error} = \frac{\text{absolute prediction error}}{\text{traffic in terms of the number of cars}}.$$

Intuitively, the relative prediction error measures the average prediction error per car in the road network. At a prediction time of about 20 minutes the prediction error adds up to an average of about two cars per road segment (c.f. Figure 18.4(b)) if the complete history is used. At this time, the average number of cars per road segment is about fourteen (c.f. Figure 18.4(a)), yielding a relative prediction error of about 14%. With increasing prediction time the relative prediction error increases rapidly, e.g. at a prediction time of about 40 minutes the relative prediction error reaches about 1.5 cars average, at average of only four cars per segment.

We also measured the relative prediction error at the four most relevant road segments averaged over different prediction time intervals. These results depicted in Figure 18.5 show that the relative prediction error is between 5% and 15% for short-term predictions and between 10% and 60% for long-term predictions when taking the motion history into account.

Additionally, we measured the traffic at specific time intervals in terms of number of cars. The results are given in Table 18.1. If we compare the resulting traffic of the single road segments to the average of all relevant road segments, we can see that the road segments selected for this experiment show a heterogeneous traffic. This experiment shows that the consideration of the history of each car has significant influence on the prediction quality for short-term predictions (about 20 minutes prediction) as well as for long-term predictions (about 40 minutes prediction).

In the next experiment we evaluated how the length of the history which was taken into account for each car influences the prediction quality. For this experiment we have run the simulation with a set of 500 cars. We measured the average prediction error for

Road Segment Id	$0-\infty$	18-22 min	38-42 min
4	11.97	16.93	3.00
17	24.94	30.50	6.00
157	11.68	16.88	4.46
433	14.72	16.33	5.14
sum.	142.84	257.19	74.12

Table 18.1: Traffic table for the experiments shown in Figure 18.5.

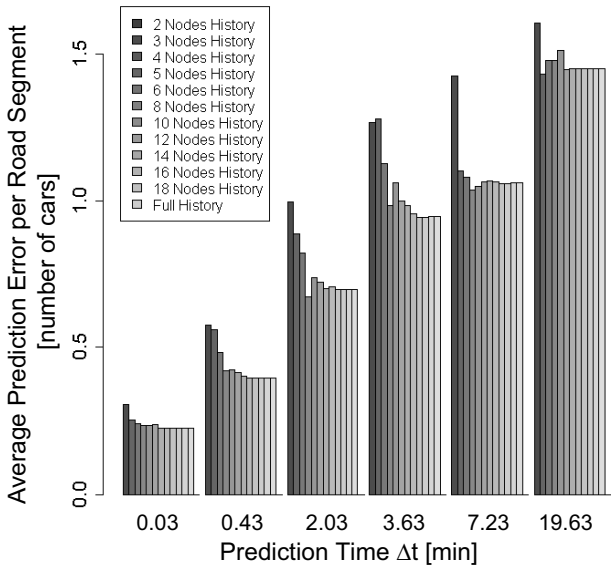


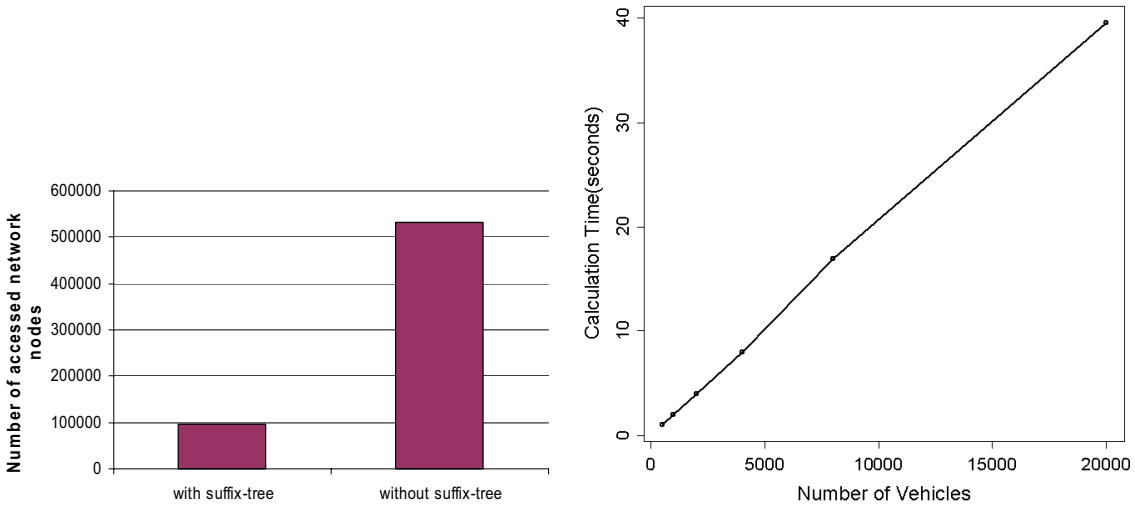
Figure 18.6: Average prediction error for varying motion history.

several prediction times by varying lengths of observed histories. The results are depicted in Figure 18.6. An interesting observation is that the length of the history in terms of passed nodes has similar effect for short-term predictions and long-term predictions. A history longer than ten nodes does not make any difference in the prediction error.

18.5.2 Experiments Concerning the Efficiency

In the next experiments we show the performance comparison between the proposed prediction strategies. In particular, we compare the first solution where the future path probabilities for each car are computed at run-time with the approach using the pre-computed suffix-tree. The performance is measured in terms of the number of network nodes which have to be accessed to predict the traffic density at each road segment for one certain point of time in the future. Additionally we measured the absolute runtime required to make the prediction. The results of the number of accessed network nodes are shown in Figure 18.7(a).

Furthermore, we evaluated the scalability of our traffic prediction approach when using



(a) Accessed Network Nodes (b) Runtime

Figure 18.7: Performance of the traffic density prediction.

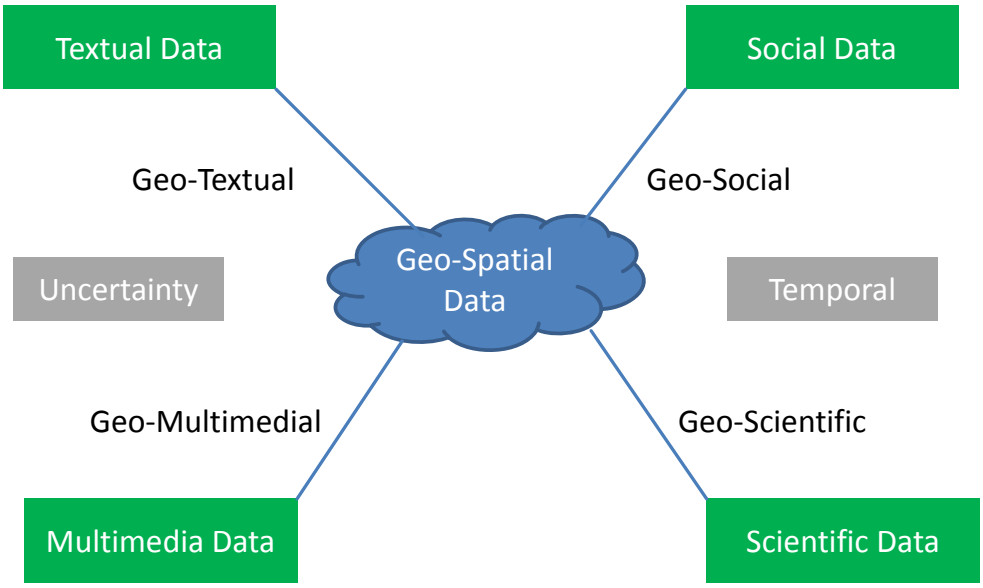
the suffix-tree in order to accelerate the prediction. Figure 18.7(b) demonstrates the time required to make a 4-minute traffic-prediction for the entire road network. We measured the runtime for varying number of cars in the traffic network. Obviously, the prediction runtime increases linearly with increasing number of moving objects.

18.6 Conclusions

In this chapter, we proposed an approach for density prediction in traffic networks. We introduced a statistical model that is used to predict the traffic density on any edge of the network at some future point of time. Furthermore, we showed how short-term observations can be used to improve the prediction quality and how the traffic densities can be computed in an efficient way. We experimentally demonstrated that our approach achieves high prediction qualities in particular when taking the history of the moving individuals into account. However, we observed that only a quite small history suffices to reach adequate prediction qualities for both short-term and long-term predictions. In addition, our runtime experiments showed that the computation of the traffic predictions can be made in reasonable time. In the future, we plan to extend our statistical prediction model by taking further observable or even learnable motion parameters into account.

Part VI

Future Visions



This part will identify research gaps still open, and provide initial ideas of how to close these gaps. Further research directions based on the work presented in this theses are given.

- Chapter 19, presents a first step towards **managing, querying and mining spatially extended spatio-temporal objects**, i.e., polygons or point-clouds moving through time while changing their location and shape. Applications for such data include meteorologic data, where spatially extended spatio-temporal objects may describe areas of low pressure, hurricanes and cyclones. Part of this vision is, to combine techniques from the field of spatio-temporal database management, with sophisticated meteorological models, to **improve the ability to predict** the occurrence as well as the movement pattern of storms. Also, mining such data may yield **interesting new patterns**, indicating environmental change, such as increasing frequency of storms.

- In modern applications, geo-spatial data is often enriched by a variety additional data sources or contexts such as social data, text, multimedia data and scientific measurements. Such data, called **multi-enriched geo-spatial data**, provides a tremendous potential of discovering new and possibly useful knowledge. The novel research challenge is to search and mine this wealth of multi-enriched geo-spatial data. The **ultimate goal** of this project is to develop a **general framework of methods** for **searching** and **mining** multi-enriched geo-spatial data in order to fuel an advanced analysis of big data applications beyond the current research frontiers. This vision is elaborated in more detail in Chapter 20.

Chapter 19

Probabilistic Ranking in Fuzzy Object Databases

As described in the introduction of this thesis, a traditional (certain) spatial database consists not only of points, but also of polylines and polygons. In this thesis, all presented techniques have been tailored to uncertain point data only. The assumption, that uncertain objects are point objects, having a single position in space that is uncertain, is valid in all of the application described so far in this thesis. In particular in geo-social network applications, the uncertainty component is the position of users. The positions of interesting locations, such as bars, and restaurants, are generally known for certain and do not change frequently over time. Even when a bar switches to a new place, there is no uncertainty in between - there might be a time interval during which the bar is at no place at all - but it is not possible to interpolate the position of the bar in space.

Yet, there exists other applications, where uncertain objects are spatially extended. For example, satellite images of a spatially extended object, such as a building, or a cyclone. The shapes captures by such images may be uncertain, as images taken at different day times may have different light intensities, or some images may be blurred due to clouds. Furthermore, images may be obsolete such that the current position of an object, such as a cyclone, has to be extrapolated given its recent locations and meteorological movement models.

Clearly, managing, querying and mining such objects is a big challenge, since we have seen that even the simpler case of managing, querying and mining uncertain point data is already a non-trivial tasks. Nevertheless, we were able to make a first big step towards efficient management of uncertain spatially extended uncertain data published at the 21st ACM International Conference on Information and Knowledge Management (CIKM) 2012 as a poster paper ([24]). This work is presented in the following chapter.

This work uses an existing model, called *fuzzy objects model* to represent spatially extended uncertain objects. It gives a first solution to the problem of probabilistic similarity ranking on such objects. This solution applies the paradigm of equivalent worlds, in order to transform spatially extended uncertain objects to uncertain point objects, such that the distribution of the probabilistic distance of this uncertain point object is equal to

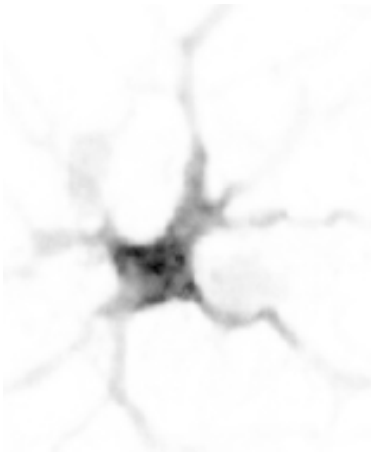


Figure 19.1: A typical cell image in biomedical analysis. Darker pixels have higher probability of belonging to the cell [215].

the distribution of the probabilistic distance of the original uncertain spatially extended object. After this transformation, existing approaches for probabilistic similarity ranking on uncertain point objects (see Chapter 7) can be applied to solve the problem. This first step towards efficient querying of uncertain spatially extended uncertain objects is presented in the next chapter.

19.1 Introduction

Nowadays, as satellites are producing images much faster than before, the huge size of the datasets rules out the approach of identifying objects and relationships manually. Instead, we must rely on automatic techniques. However, it is often impossible to interpret the objects in satellite images unequivocally due to the limitation of image resolution and due to weather effects. In order to reflect the uncertainty embedded in images and to offer subsequent analysis more information to work with, a *probabilistic mask* is produced on the extent of identified cells by probabilistic segmentation [132].

For example, Figure 19.1 shows a typical cell image in biomedical analysis. The boundary of the cell cannot be identified easily, i.e., it is not crisp. Under the model of probabilistic masks, different pixels in the image will be assigned different probabilities to indicate the likelihood that the pixel belongs to the cell. By this means, each object is transformed into a collection of points with probabilities. As such, uncertainty lies in their compositions, i.e., a point may or may not belong to the object. Therefore, they are essentially different from the uncertain databases in which the objects are assumed to have probabilistic locations at query time. The concept of probabilistic masks essentially represents the cells in images as fuzzy objects. Although fuzzy objects have long been studied in the GIS community [165, 176, 10, 164], common spatial queries such as ranking queries still remain uninvestigated at large. In this chapter, we will address the problem of ranking objects in Euclidean space over large fuzzy datasets. This type of query has many applications in the

geo-spatial field, such as "Find all the houses in the database in the order of their distance from location p or object o" or "Find the nearest city of population greater than 30,000 to Brisbane, Australia". Also, such queries are of great interest in the biomedical field such as brain Alzheimer's disease analysis [149].

Before stepping further to propose new solutions to rank objects in a fuzzy spatial database, we have to raise the question: does the fuzziness of objects change the nature of traditional probabilistic ranking of uncertain objects? Or can we adopt existing solutions to support this type of data? If we want to plug the fuzzy type into existing probabilistic ranking algorithms, an object should be represented by a single point in every possible world. Then a problem arises: how to find this point? Usually it is not easy to choose a suitable representative point because the underlying object (e.g. neuron cells) has a complex shape, and not all parts of the object are equally important (kernel vs. boundary) in terms of their confidence. Hence, choosing arbitrary points will cause considerable information loss and even produce misleading results.

This chapter shows how this point, which defines the distance between a query point q and a fuzzy object A can be found dynamically, for each world of A. This observation will allow to map each fuzzy object A (which generally consists of many points in a possible world) to a probabilistic object $\mathcal{A}$ (which consists of alternatives of which at most one holds in a possible world). The mapping is done in a way such that a fuzzy ranking performed in the fuzzy space is identical to a probabilistic ranking performed in the mapped probabilistic space. Then, we can directly apply existing solutions for efficient ranking of probabilistic objects.

19.2 Preliminaries

This section introduces the notations used in the remainder of this chapter. Some notions may be inconsistent with existing works, which however cannot be avoided, since this chapter uses notations from both the field of fuzzy data management and probabilistic data management, and disambiguation is needed for some notations.

19.2.1 Fuzzy Objects

Our definition of a fuzzy object is based on fuzzy sets [211]. A fuzzy object A consists of a set of instances and a membership function $\mu_A : \mathbb{R}^d \rightarrow [0, 1]$ mapping each instance $a \in A$ to a real value $\mu_A(a)$ within the interval $[0, 1]$. Values in between are associated with the fuzzy member instances of the object. In this work, we assume the discrete case, where each object A_i of a fuzzy database $\mathcal{DB} = \{A_1, ..., A_n\}$ consists of a finite number of of multi-dimensional points.

Definition 71 (Fuzzy Object)**.** *A fuzzy object* $A = \{\langle a \in \mathbb{R}^d, \mu_A(a)\rangle | \mu_A(a) > 0\}$ *in the d-dimensional space is represented by a set of spatial points a, each associated with a* membership probability $\mu_A(a)$ *which indicates the probability of a belonging to A. The function* μ_A *is called* membership function *of A.*

(a) $\alpha = 0.1$ (b) $\alpha = 0.3$ (c) $\alpha = 0.9$

Figure 19.2: Fuzzy object for different values of α.

Any point $a \in A$ with $\mu_A(a) = 1$ is known to exist for certain, i.e., in every possible world. The largest set of such points is called the *core* of A. Any point a with $\mu_A(a) = 0$ cannot possibly belong to A. Such points do not have to be stored explicitly in $\mathcal{DB}$, as they cannot appear in any possible world. Thus, we can assume without loss of generality that $\mu_A(a) > 0$ for all $A \in \mathcal{DB}$.

To materialize a fuzzy object A into a possible world, a probability threshold α is chosen uniformly in the interval $[0, 1]$, and each point $a \in A, \mu_A(a) \geq \alpha$ belongs to A in this world, while the remaining points do not exist (cf. Figure 19.2).

Definition 72 (α-cut)**.** *Given a fuzzy object A, the sets $A_s = \{a \in A | \mu_A(a) > 0\}, A_k = \{a \in A | \mu_A(a) = 1$ and $A_\alpha = \{a \in A | \mu_A(a) \leq \alpha\}$ are called the support set, the kernel set and the α-cut of A, respectively.*

With a probability of $1 - max_{a \in A}(\mu_A(a))$, object A contains no points and, thus, does not exist.

To materialize a fuzzy database into a possible world, there exist two approaches [32]. The first approach chooses a *global* value for α, which is applied to materialize all fuzzy objects in $\mathcal{DB}$. The *local* approach chooses an individual value α_A for each object $A \in \mathcal{DB}$.

The global approach has applications in bio-medical imaging, where it can be assumed that all object images are taken by the same instruments (e.g. the same microscope). In such an application, the only unknown (and thus random) variable of the system is the intensity of any pixel required to belong to its object. The local approach has manyfold applications in geo-spatial databases, where different object images may be taken in a different environment. For instance, images taken at different day times may have different light intensities, or some images may be more blurred due to clouds. Thus, the required intensity of pixels to belong to an object may differ between objects. In this work, we will focus on the local approach, as it is a generalization of the global approach. The set of

possible worlds of an object A is given as

$$pw(A) = \{A_\alpha | 0 < \alpha \leq 1\}$$

Assuming that the set of points belonging to object A is finite, each possible world of A is associated with a non-negative probability

$$P(w_A \in pw(A)) = argmax_\alpha\{A_\alpha = w_A\} - argmin_\alpha\{A_\alpha = w_A\},$$

the set of possible worlds of $\mathcal{DB}$ is given as

$$pw(\mathcal{DB}) = \{pw(A_1) \times ... \times A_n\},$$

where is possible world is associated the probability

$$P(w \in pw(\mathcal{DB})) = \prod w_A \in pw(\mathcal{DB})P(w_A).$$

Since, unlike a probabilistic object, a fuzzy object may consist of more than one point in space, we further need to define a distance function between point sets that will be used for ranking. Here, we use the definition of [215] to define the $\alpha - distance$ between a query point and a fuzzy object.

Definition 73 (Fuzzy Distance)**.** *For a crisp query object q and a fuzzy object A, their distance in a possible world, where A is cut at α, is given by:*

$$d_\alpha(q, A) = min_{a \in A_\alpha}[d(q, a)],$$

where $d(x, y)$ is the Euclidean distance between d-dimensional vectors.

In this chapter, we focus on the issue of obtaining fuzzy objects in their order of distance from a given query point (termed ranking). This issue is of primary interest in a spatial database although it is also used in other database applications including multimedia indexing [115], CAD, and molecular biology [105]. The desired ranking may be full or partial (e.g., only the first k objects).

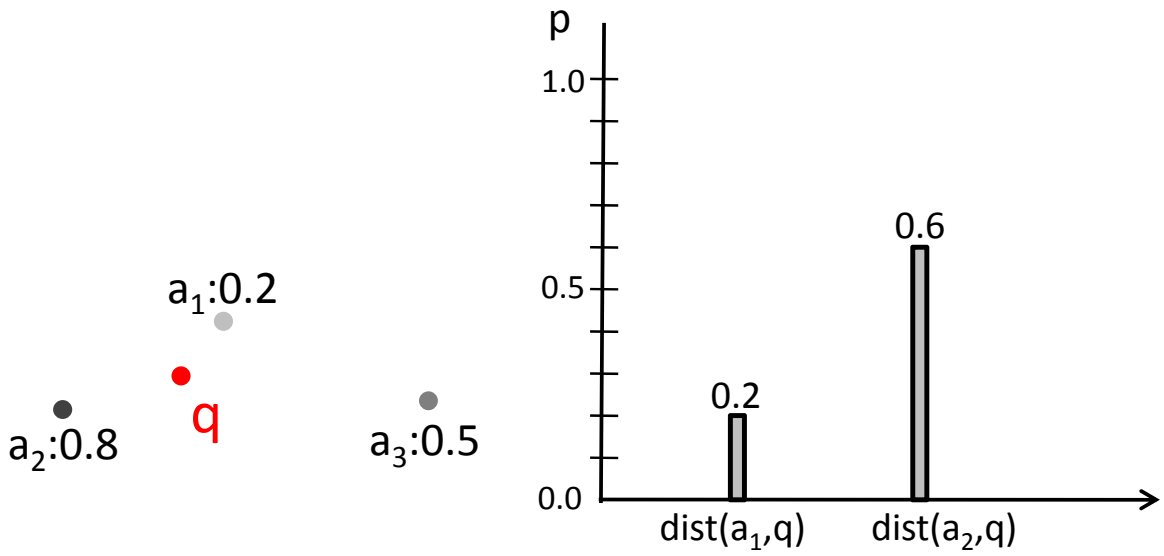
19.3 Fuzzy Ranking

The main challenge of this work is to compute, given a query point q, a probabilistic distance ranking of all database objects. That is, for each database object $o \in \mathcal{DB}$ and each ranking position $1 \leq k \leq k_{max}$, the task is to compute the probability that o is the kth closest fuzzy object to q according to the fuzzy distance function defined in Definition 73. Formally:

Definition 74. *A probabilistic ranking query $ProbRank(q)$ on a fuzzy object database $\mathcal{DB}$ returns for each object $o \in \mathcal{DB}$ and each ranking position $1 \leq k \leq k_{max}$ the probability*

$$P(q, o, k) = \sum_{w \in pw(\mathcal{DB}), Rank(o)=k} p(w)$$

that o is on rank k with respect to the distance to q.



(a) $d_\alpha(q, A)$. (b) Distance PDF.

Figure 19.3: Translation from fuzzy to probabilistic distance.

This section first gives an intuition the main idea of translating fuzzy objects into probabilistic objects, equivalent with respect to the query object and the ranking query predicate.

Then, this intuition will be formalized into a translation algorithm. The result of this algorithm can then be given to a probabilistic ranking algorithm.

19.3.1 Identifying the Distance Representative

Essentially, the fuzzy distance (cf. Def. 73) corresponds to the minimal distance of all pairs of points existing in the corresponding α-cut. Thus, in a possible world (i.e., for some α-cut), the distance between the crisp query object q and a fuzzy object A is defined by the distance between q and a point $a \in A$. All other points $p \in A \setminus \{a\}$ can be completely ignored for the distance computation. This important property will be exploited in the algorithm to be presented in this chapter. It allows to treat possible worlds having the same distance representative as equal, thus applying the paradigm of equivalent world (see Chapter 3).

Example 31. *Figure 19.3(a) illustrates this observation for the distance between the query point q and a fuzzy object A. A consists of 3 instances a_1, a_2 and a_3, with membership probabilities $\mu_A(a_1) = 20\%$, $\mu_A(a_2) = 80\%$ and $\mu_A(a_3) = 50\%$. a_1 is closest to q, and, thus, in any possible world where a_1 exists, a_1 defines the fuzzy distance between q and A. Since a_1 exists with a probability of 20%, we can conclude that a_1 defines the distance between q and A with a probability of 20%. Instance a_2 is the next-closest point to q, and exists with a probability of 80%. By Def. 72, and since $\mu_A(a_1) \leq \mu_A(a_2)$, it holds that a_1*

Algorithm 12 Object Transformation: Fuzzy to Probabilistic

Require: Point q, Fuzzy Object $A = \{a_1, ..., a_n\}$
1: $\mathcal{A}^q$ = new ProbabilisticObject
2: FLOAT $maxProb$
3: **for** $a_i \in A$ **do**
4: a_i^q = new ProbabilisticInstance
5: $a_i^q.coords = a_i.coords$
6: $p(a_i^q) = 0$
7: **if** $\mu_A(a_i) > maxProb$ **then**
8: $p(a_i^q) = \mu_A(a_i) - maxProb$
9: $maxProb = \mu_A(a_i)$
10: **end if**
11: $\mathcal{A}^q = [\mathcal{A}^q, a_i^q]$
12: **end for**
13: return ($\mathcal{A}^q$)

can only exist if a_2 also exists. Thus, the random event "a_2 exists and a_1 does not exist" has a probability of $80\% - 20\% = 60\%$. If this event holds, then a_2 defines the distance between q and A. Finally, instance a_3 is farthest away from q. Due to Def. 72, and since $\mu_A(a_3) \leq \mu_A(a_2)$, a_3 exists only if a_2 exists. And since a_2 is closer to q than a_3, it is not possible for a_3 to define the distance between q and A in any possible world. With a probability of $1 - max_i(\mu_A(a_i)) = 20\%$ object A has no instance at all, and, thus, does not exist.

Due to the observations made above, we will now rewrite A as a probabilistic object $\mathcal{A}^q$, consisting of two probabilistic instances a_1^q and a_2^q having probabilities $p(a_1^q) = 0.2$ and $p(a_2^q) = 0.6$. The spatial locations of a_1^q and a_2^q correspond to the locations of a_1 and a_3, respectively. Note that there is no instance a_3^q, since a_3 never defines the distance between q and A and thus, can be ignored. Now, keeping in mind that instances of a probabilistic object are mutually exclusive, we trivially get that instance a_1^q defines the distance between q and $\mathcal{A}^q$ with a probability of 0.2, while instance a_2^q defines this distance with a probability of 0.6. $\mathcal{A}^q$ does not exist at all with a probability of $1 - \sum_i p(a_i^q) = 0.2$. In summary, we get a probability density function (PDF) of the distance between q and A as depicted in Figure 19.3(b).

In this example, we could guarantee that by construction of the probabilistic object $\mathcal{A}^q$, the probability of a fuzzy instance $a_i \in A$ defining the fuzzy distance between q and A is equal to the probability that the corresponding probabilistic instance a_i^q exists. In the next section, we will formalize this approach and show that a probabilistic ranking on the translated probabilistic objects, yields the same result as a fuzzy ranking on the raw fuzzy objects.

19.3.2 Translation to Probabilistic Objects

Consider Algorithm 12, which takes as input parameters a (crisp) query point q and a fuzzy object $A = \{a_1, ..., a_n\}$. Let μ_A be the membership function of A. Without loss of generality, assume that the instances $a_i \in A$ are sorted increasingly in their distance to q.[1] The main goal of the algorithm, is to transform a fuzzy object into an uncertain objects defined according to the X-tuple model described in Part 2.2. The algorithm iteratively creates a probabilistic instance a_i^q for each fuzzy instance a_i of A. The spatial coordinates of the probabilistic instances are set to the spatial coordinates of the corresponding fuzzy instances. The probability of each probabilistic instance is set to the difference between the corresponding fuzzy instance and the maximum membership probability of all fuzzy instances seen so far. All the newly created probabilistic instances are then unified into a single probabilistic object $\mathcal{A}^q$. The later step again applies the paradigm of equivalent worlds, by unifying worlds which are equivalent for a ranking query having the given query point.

Lemma 43. *After running Algorithm 12 with parameters A and q, it holds for each fuzzy instance $a_i \in A$, that the probability that a_i defines the fuzzy distance between q and A, equals the existential probability $p(a_i^q)$ of instance a_i^q of the returned probabilistic object $\mathcal{A} = \{a_1^q, ..., a_n^q\}$.*

Proof. By induction: Instance a_1 exists with a probability of $\mu_a(a_1)$, and in any world where a_1 exists, it must define the fuzzy distance between q and A. Thus a_1 defines the distance between q and A with a probability of $\mu_a(a_1)$, and $p(a_1^q) = \mu_a(a_1)$.

An instance $a_i, 2 \leq i \leq n$ defines the distance between q and A if and only if instance a_i exists, and no closer instance $a_j, j < i$ exists. This is clear due to the assumption that the instances are sorted, and thus the set $\{a_j\}_{j<i}$ corresponds to the set of all instances of A which are closer to A than a_i. By definition of an α-cut, a_i exists if and only if $\alpha < \mu_A(a_i)$. Furthermore, no point in the set $\{a_j\}_{j<i}$ exists if and only if $\forall j < i : \alpha > \mu_A(a_j)$, i.e., if and only if $\alpha > max_{j<i}(\mu_A(a_j)$. Since α is the realization of a uniformly $[0, 1]$ distribution, the probability of the random event $max_{j<i}(\mu_A(a_j) < \alpha < \mu_A(a_i)$ equals $\mu_A(a_i) - max_{j<i}(\mu_A(a_j)$ if $\mu_A(a_i) > max_{j<i}(\mu_A(a_j))$ and 0 otherwise. In line 8 of Algorithm 12, the value of $p(a_i^q)$ is assigned to this probability. □

In a nutshell, Lemma 43 allows us to transform a fuzzy object A into an uncertain object $\mathcal{A}$ defined according to the X-Tuple model. For disambiguation, uncertain objects will be denoted as probabilistic objects.[2]. The resulting probabilistic object $\mathcal{A}$ is defined such that the probability of a probabilistic instance $a_i^q \in \mathcal{A}$ corresponds to the probability that the instance $a_i \in A$ defines the distance between q and A.

[1] We make this assumption to ease readability by avoiding double indices. If instances a_i are not sorted, we can simply sort them a priori in log-linear time.

[2] Since fuzzy objects are uncertain, too.

Corollary 12. *The probability density function of the fuzzy distance between q and A equals the probability density function of the probabilistic distance between q and $\mathcal{A}$.*

Proof. This corollary follows from the construction of $\mathcal{A}$. Since each fuzzy instance $a_i \in A$ has the same spatial attributes as its corresponding probabilistic instance $a_i^q \in \mathcal{A}$, it also holds that their Euclidean distances are equal, i.e., $d(q, a_i) = d(q, a_i^q)$. Furthermore, the probability of each fuzzy instance $a_i \in A$ of defining the distance to q, equals the probability $p(a_i^q)$ of its corresponding probabilistic instance $a_i^q \in \mathcal{A}$ to define the distance to q. Thus, both probability density functions, have the same probability values at the same distance values, and are thus equal. □

Corollary 12 directly leads to the following observation:

Corollary 13. *Let $\mathcal{DB}$ be a fuzzy database, and let q be a query point. Let $\mathcal{DB}$ be a probabilistic database obtained by transforming all fuzzy objects in $\mathcal{DB}$ using Algorithm 12. It holds a fuzzy distance ranking on $\mathcal{DB}$ using query object q is equal to a probabilistic distance ranking on $\mathcal{DB}$ using query object q.*

Proof. The above corollary follows directly from Corollary 12, since the ranking is performed on the distance objects. □

Finally, Corollary 13 allows us to plug in existing algorithms for probabilistic ranking (e.g. those presented in and described in the related work section of Chapter 7 which run in $O(N \cdot log(N) + N \cdot k)$, where N is the total number of instances in the database, and k is the length of the requested partial ranking.

The transformation to probabilistic objects requires the instances of each object to be sorted w.r.t. their distance to the query object, which is performed in $O(|\mathcal{DB}| \cdot N \cdot log(N)) = O(N \cdot log(N))$ runtime. This additive component has no affect on the total asymptotic run time of $O(N \cdot log(N) + N \cdot k)$ of the probabilistic ranking.

A total runtime of $O(N \cdot log(N) + N \cdot k)$ is acceptable in probabilistic ranking settings, where the number of possible alternatives per object is usually rather small (less than a thousand).

19.4 Conclusions

To the best of our knowledge, this is the first approach for probabilistic ranking of fuzzy objects according to possible world semantics. The main idea is to transform each fuzzy object into a probabilistic object, such that a probabilistic ranking performed on these objects is guaranteed to be equal to a probabilistic ranking on the raw fuzzy objects. Currently, the algorithm requires a full scan of the database. Therefore, we are currently working on an implementation of an aggregate R-tree [118] to index all points of a fuzzy object in order to minimize the number of points that have to be accessed at query time. Furthermore, we see a lot of potential in optimizing Algorithm 12 by adapting techniques for dynamic skyline search [46], as we note that the set of points of a fuzzy object having a probability greater zero in the transformed probabilistic space, equals the dynamic skyline of query object q in the distance-probability space. A third extension would extend the current solution to fuzzy query objects. Another part of our future work is to consider fuzzy objects with continuous extent in space. Transforming such objects into the probabilistic object space will yield continuous distance PDFs, which can be ranked using the approach proposed by [124].

19.5 Research Directions

The problem of querying fuzzy data is at least as hard as querying probabilistic data. Similar to probabilistic objects, the number of possible worlds of a fuzzy database is exponential in the number of fuzzy database objects, since there may be one possible worlds for each combination possible shapes of fuzzy objects. Furthermore, in each possible world, a fuzzy object is no longer described by a single point, but rather by a, generally large, set of points. This makes the problem of querying fuzzy objects at least as hard as querying probability objects. Nevertheless, this chapter has shown that efficient query processing is possible, by efficiently answering an important class of spatial queries, the class of probabilistic ranking queries. This first step opens up a number of further research paths. While other types of spatial queries, as defined in Chapter 1 require novel solutions, other problems, such as shape-based similarity between fuzzy objects need to be studied. This problem requires to find, for two given fuzzy objects, the probability distribution of the similarity between these objects, where similarity is defined by existing similarity measures for spatially extended (certain) objects. Also, models are required to capture the change, in location and shape, of fuzzy objects over time. Here, a future vision is to co-operate with geographers, geologists and meteorologist to provide domain specific models, for example to model the past and future motion of a cyclone between observations. Such a project may require to adapt the techniques presented in Part V, to model the location, as well as the shape of a fuzzy object between discrete observations.

Chapter 20

Semantically Rich Geo-Spatial Data

In modern applications, spatial and spatio-temporal data is enriched by multiple additional sources or contexts such as social data, text, multimedia data and scientific measurements, called multi-enriched geo-spatial data. A novel research challenge is to search and mine this wealth of multi-enriched geo-spatial data. Recent state-of-the-art methods for analyzing enriched geo-spatial data are limited to an enrichment by at most one additional data source. In the future, we envision to develop new techniques manage, query and mine multi-enriched geo-spatial data. Such techniques will again need modern techniques to properly handle the uncertainty as well as the temporal variability of such data.

20.1 Overview

The four classes of context which are most relevant to current applications and are described in the following.

- **Social data** contains information about real-life social activities of people, such as friendship relations, shared activities and common interests. The dimension of location brings social networks back to reality, bridging the gap between the physical world and online social networks.

- **Textual data** consists of raw text information, such as user reviews, news articles and textual descriptions of objects. This data can contain textual location information, or it can be tagged with geographic information. Explicitly using this location information permits to find information that is not only related to the textual context, but also geo-spatially related.

- **Multimedia data**, particularly audio-visual and image data, has become ubiquitous as millions of pictures and video files are shared every day. While the extraction of content information from multimedia data has been well studied, we need to explore new possibilities of combining this content and geo-spatial information.

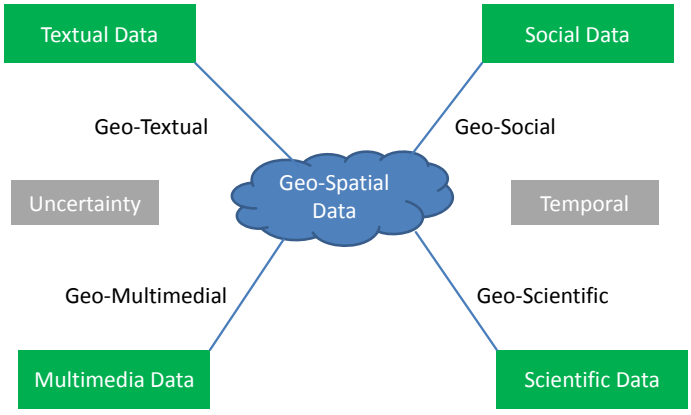


Figure 20.1: Semantically Rich Geo-Spatial Data: An Overview.

- **Scientific Measurement Data** arises in many applications where feature values are extracted from the physical world, including objects (such as archeological artifacts like bones, skeletons, coins) as well as application specific parameters (such as environmental measurements or medical data). Spatial location is an inherent but underused component of these data types.

A need for this project arises from the fact that, on the one hand, such contextually enriched geo-spatial data is collected and published ubiquitously all over the world, creating extremely large sets of data but, on the other hand, we still lack proper methods to analyze this wealth of data in order to utilize the immense amount of hidden knowledge. The collection of this data has become possible due to the widespread availability of new technologies for collecting, publishing and sharing data, such as modern sensor and communication technologies, smart mobile devices, and advances in web-based technologies. Therefore, a variety of popular applications where users can voluntarily publish geo-tagged information, leads to an explosion and continuously growing bulk of such data. For example, users can comment on an event at the exact place where the event is happening (e.g. via Twitter), voluntarily share their present location on a website (such as Foursquare) for organizing a group activity in the real world, record travel routes with GPS trajectories to voluntarily share travel experiences in an online community (for example GeoLife), or log jogging and bicycle trails for sports analysis and experience sharing (as on Bikely). Obviously, this data contains an incredible wealth of information that could be discovered with appropriate data analysis techniques, in order to improve other scientific disciplines or simply everydays life. While analyzing single-enriched geo-spatial data, i.e., data enriched by only one additonal context, is already a challenge by itself, considering several additional contextsi.e., dealing with multi-enriched geo-spatial data obviously provides an even more significant gain for data analysis applications since it will reveal a new level of knowledge that could not have been derived before.

As an example application where multi-enriched geo-data can be used for the benefit of all, consider the application “Google Flu Trends”. Google Flu trends helps to monitor a disease such as influenza (flu) or dengue fever using aggregated Google search data of

related search terms, as well as location information about the origin of these searches. It estimates and predicts flu activity closely matching clinical data, but preceding it by 1-2 weeks, and, thus, gives an extra time for taking precautious measures due to the use of geo-spatial data enriched by textual information, i.e., geo-textual data. Furthermore, individuals can use information about flu activity in their spatial vicinity to estimate their probability of catching a flu, in order to take appropriate actions. However, with multi-enriched geo-spatial data, it is possible to go even further, e.g. by additionally integrating geo-social context and geo-scientific context. When people catch a flu, they may publish this information including their symptoms in one of their social networks. This information can be used to adapt the a-posteriori infection probability of friends they recently met. Additional data from medicine, such as incubation time and infectious period correlated with the published symptoms can then be used to warn individuals having a high likelihood of being or becoming infected. These warnings will not only allow these individuals to take actions to alleviate their disease and possibly even prevent a breakout. Furthermore, these warnings will give individuals an awareness of their condition in order to prevent infection of their friends, thus breaking the chain of infection.

20.2 Research Directions

In the future, more applications will arise where multi-enriched geo-spatial data will emerge in big quantities. In all of the applications, the ability to properly handle the aspects of uncertainty and temporality of such data will be paramount. Temporal information is inherently tied to spatial information in modern geo-spatial data, as geo-tags associated with any kind of semantic data are always associated with a time stamp. Uncertainty is not only a direct consequence of temporal information, and thus the presence of obsolete and outdated information, but it is also a direct result of the process that generates such data. data obtained by crowd-sourcing is inherently uncertain, as individual humans may give incorrect information, due to lack of better knowledge, or deliberately due to lack of time or interest. Next-generation networks ([120]) have inherent uncertainty, due to measurement errors leading to contradictory data.

For this reason, the research field of managing and mining multi-enriched geo-spatial data is a natural follow-up to the research presented in this thesis. In order to achieve the goal of successfully query and mine multi-enriched geo-spatial data a series of subtask has to be solved:

- **Modeling Multi-Enriched Geo-Spatial Data.** In subtask, we have to cope with both the heterogeneity and uncertainty of contextually enriched geo-spatial content, as such data is often derived from a large variety of different sources and contexts, including social information, textual descriptions, images, sensor and other scientific measurements, trajectories in space and time and many others. We will develop techniques to synchronize heterogeneous data sources that may be inconsistent, as most sources of geo-spatial data are inherently uncertain, due to impreciseness of sensing devices, due to a possible obsoleteness of information, and due to human errors which are ubiquitous in user-generated content. In this subtask, this uncertainty will be addressed by exploring new directions for handling uncertainty effectively and efficiently in order to avoid inaccurate and possibly wrong results. In addition to the heterogeneity of sources, a further challenge is to cope with different types of data. There will be a crucial need for new algorithmic approaches to search in geospatial data that also contains textual information (i.e., geo-textual data), e.g. modelled as a high-dimensional term frequency vector, temporal information (i.e., spatio-temporal data), e.g. modelled as a time series, social information of users (i.e., geo-social data), e.g. modelled as a graph, and more. This subtask requires to find a suitable data representation to maximize the amount of useful information captured. Finding proper data models and choosing smart strategies for feature selection are vital requirements to store information in a concise way, in order to efficiently and effectively describe, induce and explore information. Once suitable models to describe the data have been developed, it is crucial to define distance measures, in order to determine functions to measure similarity and dissimilarity for contextually enriched spatial objects. For example, we need to cope with questions like which pair of individuals is more similar? The answer to this question could e.g. be a pair of individuals having

a spatial distance of 10 km and a social distance of 2 (as defined by the shortest path in some given social network), or a pair having a spatial distance of 20 km and a social distance of 1. Thus, the main challenge when deciding which pair is the proper answer for a given query is to combine spatial- and semantic distance (or similarity) measures in a meaningful way.

- **Querying Multi-Enriched Geo-Spatial Data** Based on sufficient techniques for modeling contextually enriched geo-spatial data, we can define useful similarity queries such as ϵ-range queries, kNN-Queries, RkNN queries and ranking queries. An example of such a query using geo-textual data is *return the five news articles most relevant to the keywords Car Theft in my spatial proximity (i.e., having a spatial distance of less than ϵ to my house).* Another example using geo-social data is *return a group of five people in my close vicinity that are strongly connected in a social graph, in order to advertise a group offer such as a free restaurant table to this group.* An example combining more than one semantic data source is a query such as *give me a warning if I have recently been close to friends that has published information that he now shows symptoms of the flu.*.

  We need a semantically meaningful approach of combined query evaluation that combine spatial and semantical features. Such a query evaluation to develop universal index structures that can handle uncertain spatial data as well as all different combinations of non-spatial data while providing good performance at the same time.

- **Mining Multi-Enriched Geo-Spatial Data** Mining contextually enriched geo-spatial data poses another set of very interesting challenges. While the objectives in data mining are generally quite diverse, different paradigms including clustering, outlier detection, regression, classification, and frequent pattern mining all hunt for rather diverse types of patterns. The methods on which we are focusing here share many common basics. For example, most data mining algorithms assume vectorized data or are based on distance computations and can be accelerated by an index that supports similarity queries.

We envision this research direction to be the first to analyze multi-enriched geo-spatial data, and thus will ultimately bridge the gap between the present capabilities of data acquisition and the currently underdeveloped abilities to analyze and utilize this data to benefit science, industry and everyday life.

Part VII

Summary

Tutorial on Managing and Mining Uncertain Spatial and Spatio-Temporal Data

One of the main goal of this thesis is to give a thorough introduction to the field of uncertain data management in Part II. For this purpose, a tutorial to this the field of uncertain data and uncertain spatial data management has been presented in Part I and Part II of this thesis. These parts contain a significant update and extension of our VLDB tutorial [157]. These sections are aimed at a graduate student level, to give a jump start to the field, and to provide all the necessary basics to quickly follow and understand existing publications. The knowledge of current publications is mandatory to gain a deep knowledge of the problems and solutions of this field. This knowledge is the basis to develop new ideas, to push the field further forward. In the short term future, we plan to use the ideas of Part I and Part II of this thesis to compile a new conference tutorial, which will include all of the core models, problems, ideas and techniques of the old tutorial [157], but also includes new concepts that have only been presented in this thesis. This new tutorial will also be extended to handle uncertainty in spatio-temporal data.

The Paradigm of Equivalent Worlds

In the last years, a large number of efficient solutions for a variety of querying and data mining problems on uncertain data have been proposed throughout the community. All the works solve a common problem: Given an exponential large number of possible worlds, give a solution that runs in polynomial time. To solve these problem, all these works, implicitly or explicitly partition the set of all possible worlds into sets of worlds, that are equivalent with respect to the given query predicate. The query is then performed on these partitions only, without the enumeration of each individual world. The paradigm presented in Chapter 3 formalizes this approach, and identifies requirements which must hold in order for this paradigm to be applicable. This formalization, helps to give an intuition to quickly find an efficient solution for a problem that has an efficient solution. For problems not having an efficient solutions, the formalized requirements help to identify these problems.

Querying Uncertain Spatial Data

A research field that has been covered thoroughly in Part III of this thesis is the problem of querying uncertain spatial data. In this part, efficient solutions for the most relevant types of probabilistic queries on uncertain spatial data have been presented. These solutions follow the presented in Chapter 3. In particular, and efficient spatial pruning approach for rectangles has been proposed in Chapter 5. This spatial pruning approach is enriched by a novel probabilistic pruning approach to give an efficient solution for the problem of k-nearest neighbor search on uncertain spatial data, presented in Chapter 6. The problem of probabilistic similarity ranking in uncertain spatial data has been covered in detail

in Chapter 7, presented the first solution running loglinear in the size of the uncertain database. Previous solutions have quadric, cubic or exponential run-time. Finally, efficient solutions to the problem of reverse k-nearest neighbor queries are presented in Chapter 8.

There exists more spatial query types, such as the spatial skyline query, which has many applications in decision making problem. For this problem, efficient solutions for uncertain data have been proposed by the community ([205, 148]). Yet, it might be possible to improve these solutions by applying the paradigm presented in Chapter 3. Apart from additional query types, from our point of view, the field of querying uncertain spatial data is solved to a large degree - as the most relevant spatial query types have been covered. The new challenge is to apply, adapt, and improve these solutions for new research fields, such as querying spatio-temporal and querying multi-enriched geo-spatial and multi-enriched spatio-temporal data.

Mining Uncertain Spatial Data

A first efficient solution for a special case of the problem of spatial co-location mining in uncertain spatial data has been presented in Part IV of this thesis. This special case, where the neighborhood of spatial objects is defined by its proximity to certain points of interest, is of particular interest in geo-social networks, where points of interest correspond to meeting places, like bars and restaurants. To solve this problem, two subproblems had to be solved. For the first subproblem, probabilistic instances need to be computed, that is, the probability that an object is close to a point of interested has to be computed. This can be done by perform a probabilistic range query as described in Chapter 4. The second subproblem is to find frequent items in the probabilistic transaction defined by the results of the first subproblem. For this problem, an exact solution has been presented in Chapter 10, and an approximate solution has been presented in Chapter 11.

The field of mining uncertain spatial data is by no means solved. A different data mining problem on uncertain spatial data is the problem of probabilistic spatial clustering. We have performed initial research in the field of probabilistic density based clustering of uncertain spatial data. Finding an efficient solution to this problem seems to be hard problem, due to the problem of distance dependencies explained in Example 16 and Figure 4.3. Due to this problem, the density connectivity of all pairs of objects are stochastically dependent random events, making the task of finding the probability of a cluster a hard problem. The approach proposed in [112] ignores this problem, thus yielding only approximate results. Finding an exact approach that allows probabilistic density based clustering on uncertain data is a future challenge. One step towards this challenge was made in Chapter 4, where solutions for probabilistic range queries on uncertain were given. These solutions can be applied to compute the probability that a given object is a core object, an important component of a probabilistic version of the DBScan [68] algorithm. Other data mining problems on uncertain data, such as spatial outlier detection remain unsolved for uncertain spatial data.

Modeling, Querying, Indexing and Mining Uncertain Spatio-Temporal Data

In Part V, uncertain spatio-temporal data has been modelled as a stochastic process, which is a time-dependent random variable. This is an intuitive extension of existing models of (non-temporal) uncertain spatial data that is generally modelled as a random variable. The stochastic process used in this part is a first order Markov chain, a very simple model, that assumes that the future is conditionally independent of the past, given the present. Thus, the future motion of an object only depends on it's current location, independently of where the object came from. At first, this model seems rather restrictive, since in most applications, objects move on a shortest path between their start and their destination, rather than taking random directions at each intersection until they accidently find their destination. Yet, this model becomes powerful when more than one observation is considered. When the location of an object is observed at a past, as well as a future point of time, then the initial Markov model can be adapted to account for this additional information. This adapted Markov model can effectively model the position of the object between both observations. If the object has moved on a shortest path, then the resulting Markov model will only allow shortest paths (only the shortest path if there is only one) in its model, as all other paths would take longer than a shortest path, and thus would not allow the object to reach the observed location in time. If the object does not move on a shortest path, then the adapted model gives a probability distribution over all possible detours, weighted by the probabilities of the initial Markov chain, describing empirical directions from on location to the next.

In cases where no future observation is given, no model adaption as described above can be made. This is a common case, in application where the future has to be predicted, rather than the past having to be interpolated. To handle this case, a different model has been proposed in Chapter 18. This approach uses an adaptive-order Markov chain, represented by a suffix tree. This structure stores the probabilities of going to a successor state, given the sequence of previous states visited. The initial suffix tree stores all trajectories that have been observed on a given road network. Thus, to predict the future motion of a vehicle v, given its past motion p, all past trajectories containing p, of all vehicles are selected. The future of v is predicted empirically, by evaluating the selected trajectories. Thus, if 100 trajectories contain p, out of which 60 trajectories take a right turn after p and the remaining 40 trajectories take a left turn, then v will be predicted to take a right turn with a 0.6 probability, and a right turn with a 0.4 probability. This model yields a high prediction accuracy, in particular if the model has been trained by a large number of past trajectories. However, the size of the suffix tree model is cubic in the number of trajectories, thus becoming very large. An option to reduce the size of the model, branches having a low support can be pruned. That is, trajectories that have been observed less than $minsup$ times, are removed from the suffix tree. This allows the same prediction quality for vehicles moving on a trajectory that has been observed sufficiently often in the past. However, if the vehicle moves on a rare trajectory, then a pruned branch may be reached,

allowing no prediction to be made. Thus, the minSup parameter allows to give a balance between prediction quality and space complexity. It should be noted, that branches having a low support do not provide a significant prediction in the first place. Thus, minSup should be chosen large enough to give a representative sample of past trajectories. Note that this approach is not restricted to vehicles moving on shortest paths. Instead, any empiric traffic patterns can be used to feed the suffix tree. An extension to general suffix trees, rather than shortest-path-suffix-trees has been made in [114], but is not featured in this thesis for brevity.

Final Words

In the past years, the research field of uncertain and probabilistic data management received tremendous attention from the database research community. This fact is evident, considering that all major database conferences, including ACM SIGMOD, VLDB and ICDE had at least one, usually more than one research track dedicated to this field each year. This success is impressive, considering that this research field was well ahead of its time. Modern application fields, which are as major motivation for uncertain data, have emerged only recently with the proliferation of the Web 2.0. For example, one such application includes crowd-sourced database management. Crowd-sourced data is inherently uncertain due human errors, due to deliberate errors and due to other factors, thus creating a need for techniques to handle uncertain data. Another new application field is the field of geo-social network management. In this field, both social information as well as spatio-temporal information of users is stored. Social networks may have missing or obsolete links, for which a correct prediction may improve the social network, while the uncertainty of spatio-temporal data has been motivated and discussed throughout this thesis. Solutions are required to combine both sources of information, social and spatio-temporal, to discover new knowledge and invent new and useful applications. Finally, a research field that will require sophisticated solutions to handle uncertainty, is the field of semantically rich geo-spatial data. This field, as envisioned in Chapter 20, requires a combination of data types, including social, text, multimedia and scientific data, all tied together with spatial and spatio-temporal information. All of these data types may have inherent uncertainty, due to measurement error, due to obsolete data, and due to the origin of data, often derived Web 2.0 applications.

While a trend can be observed, that the number of research sessions for the field of uncertain and probabilistic database is dropping, as most conferences only have one or two research tracks on this field, this does not mean that the number of research papers accepted in this field dropping. Rather, relevant research on the field of probabilistic and uncertain databases management is moving towards these new and emerging fields, which require solutions from this field.

In summary, the field of probabilistic and uncertain database management is relatively young, interesting and challenging field. New arising applications have already started to fuel this field, by creating new applications, keeping the research field strong as it is. How-

ever, publishing papers in this field has become much more difficult over the last decade. Initial approaches usually made very simplifying assumptions, and used rather straightforward models and solutions. Such approaches however were needed to fit their role as competitors for more sophisticated approaches. While there may still be a number of unpublished approaches, that, while simple, have a certain elegance, most newer publications in this field require an adept knowledge of the management of uncertain data. A main goal of this thesis is to impart some of the necessary knowledge to researchers that wish to join the field. If this thesis has inspired you, and if you managed to make your way through the hundred of pages, then feel free to let me know, by inviting me to a beer at any conference, or any other location that we will meet at!

Acknowledgements

The work involved for this thesis would have never been possible without the support and encouragement of many people. I want to express my gratitude to all of them, even if I cannot mention everyone here.

First of all, I would like to express my very great appreciation to the limitless and never-ending support provided by my family. Foremost thanks goes to my parents, Maria and Reinhard Züfle, who let me freely choose my way as a computer scientist, and supported this choice in every possible way. Their love has always been the greatest motivation pushing me forward on my quest. I would further like to express my very great appreciation to my grand parents, my uncles, aunts and cousins, who always welcomed me, providing a place to rest, to enjoy and ease my mind. Without all this support, this work had never been possible.
I would like to express my deepest gratitude to my supervisor, Prof. Dr. Hans-Peter Kriegel, who gave me the opportunity to work with him and his wonderful group. The atmosphere has been very inspiring for my work; the scientific discourse and the discussions with my colleagues have been very valuable for teaching and research. The family-like research environment created by Prof. Dr. Hans-Peter Kriegel created a team that never hesitates to help each other. This environment has created an upward spiral of team members helping and pushing each other forward with no day of work ever passing without a laugh, even on stressful days plagued by deadlines.

This team turned out to be friends rather than mere co-workers. Their help and support by inspiring discussions as well as humor were essential for the development of this thesis. In particular, I would like to thank PD. Dr. Matthias Renz, whose open way of expressing his thoughts is a catalyst for developing new research ideas. The collaboration with Tobias Emrich and Thomas Bernecker in this group inspired synergies that inspired a plethora of creative discussions and productive works. Susanne Grienberger deserves a special thanks. Her helpfulness and her organizational skill helped the past years to pass flawlessly. The importance of Franz Krojer, who kept our equipment running smoothly, is not to be underestimated.

I am especially thankful to Prof. Dr. Nikos Mamoulis, whose ability to quickly understand, apply and find improvements for theoretical concepts has provided me with very valuable insights and has significantly improved my work, and my skill as a researcher.

Bibliography

[1] Ust project page. `http://www.dbs.ifi.lmu.de/cms/Publications/UncertainSpatioTemporal`.

[2] Abramowitz, and Stegun. *Handbook of Mathematical Functions With Formulas, Graphs, and Mathematical Tables*, 10 ed. 1972.

[3] Achtert, E., Bernecker, T., Kriegel, H.-P., Schubert, E., and Zimek, A. ELKI in time: ELKI 0.2 for the performance evaluation of distance measures for time series. In *Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD), Aalborg, Denmark* (2009), pp. 436–440.

[4] Achtert, E., Böhm, C., Kröger, P., Kunath, P., Pryakhin, A., and Renz, M. Efficient reverse k-nearest neighbor search in arbitrary metric spaces. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Chicago, IL* (2006), pp. 515–526.

[5] Achtert, E., Kriegel, H.-P., Kröger, P., Renz, M., and Züfle, A. Reverse k-nearest neighbor search in dynamic and general metric databases. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT), Saint-Petersburg, Russia* (2009), pp. 886–897.

[6] Aggarwal, C., Li, Y., Wang, J., and Wang, J. Frequent pattern mining with uncertain data. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Paris, France* (2009), pp. 29–38.

[7] Agrawal, P., Benjelloun, O., Das Sarma, A., Hayworth, C., Nabar, S., Sugihara, T., and Widom, J. "Trio: A system for data, uncertainty, and lineage". In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea* (2006), pp. 1151–1154.

[8] Agrawal, R., and Srikant, R. Fast algorithms for mining association rules. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Minneapolis, MN* (1994), pp. 487–499.

[9] Agrawal, R., and Srikant, R. Mining sequential patterns. In *Proceedings of the 11th International Conference on Data Engineering (ICDE), Taipei, Taiwan* (1995), pp. 3–14.

[10] Altman, D. Research article. fuzzy set theoretic approaches for handling imprecision in spatial analysis. *International journal of geographical information systems 8*, 3 (1994), 271–289.

[11] Antova, L., Jansen, T., Koch, C., and Olteanu, D. Fast and simple relational processing of uncertain data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico* (2008), pp. 983–992.

[12] Arenas, M., Bertossi, L., and Chomicki, J. Consistent query answers in inconsistent databases. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (1999), PODS '99, pp. 68–79.

[13] Arenas, M., Bertossi, L. E., and Chomicki, J. Consistent query answers in inconsistent databases. In *Proceedings of the ACM International Symposium on Principles of Database Systems (PODS), Philadelphia, PA* (1999), pp. 68–79.

[14] Assfalg, J., Kriegel, H.-P., Kröger, P., and Renz, M. Probabilistic similarity search for uncertain time series. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM), New Orleans, LA* (2009), pp. 435–443.

[15] Beckmann, N., Kriegel, H.-P., Schneider, R., and Seeger, B. The R*-Tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Atlantic City, NJ* (1990), pp. 322–331.

[16] Benjelloun, O., Sarma, A. D., Halevy, A. Y., and Widom, J. Uldbs: Databases with uncertainty and lineage. In *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB), Seoul, Korea* (2006), pp. 953–964.

[17] Bentley, J. L. Multidimensional binary search trees used for associative searching. *Commun. ACM 18*, 9 (1975), 509–517.

[18] Bentley, J. L., and Douglas McIlroy, M. "Engineering a Sort Function". In *Software-Practice and Experience* (November 1993).

[19] Berchtold, S., Keim, D. A., and Kriegel, H.-P. The X-Tree: An index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB), Bombay, India* (1996), pp. 28–39.

[20] Berge, C. *Graphs and hypergraphs*, vol. 6. Elsevier, 1976.

[21] Bernecker, T., Cheng, R., Cheung, D. W., Kriegel, H.-P., Lee, S. D., Renz, M., Verhein, F., Wang, L., and Züfle, A. Model-based probabilistic frequent itemset mining. In *"Knowledge and Information Systems (KAIS). DOI 10.1007/s10115-012-0561-2."* (2012).

[22] Bernecker, T., Emrich, T., Kriegel, H.-P., Mamoulis, N., Renz, M., and Züfle, A. A novel probabilistic pruning approach to speed up similarity queries in uncertain databases. In *Proceedings of the 27th International Conference on Data Engineering (ICDE), Hannover, Germany* (2011), pp. 339–350.

[23] Bernecker, T., Emrich, T., Kriegel, H.-P., Renz, M., Zankl, S., and Züfle, A. Efficient probabilistic reverse nearest neighbor query processing on uncertain data. *PVLDB 4*, 10 (2011), 669–680.

[24] Bernecker, T., Emrich, T., Kriegel, H.-P., Renz, M., and Züfle, A. Probabilistic ranking in fuzzy object databases. In *Proceedings of the 21st International Conference on Information and Knowledge Management (CIKM), Hawaii* (2012), pp. 2647–2650.

[25] Bernecker, T., Kriegel, H.-P., Mamoulis, N., Renz, M., and Züfle, A. Scalable probabilistic similarity ranking in uncertain databases. *IEEE Transactions on Knowledge and Data Engineering 22*, 9 (2010), 1234–1246.

[26] Bernecker, T., Kriegel, H.-P., Mamoulis, N., Renz, M., and Züfle, A. Continuous inverse ranking queries in uncertain streams. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Portland, OR* (2011), pp. 37–54.

[27] Bernecker, T., Kriegel, H.-P., and Renz, M. ProUD: probabilistic ranking in uncertain databases. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China* (2008), pp. 558–565.

[28] Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., and Züfle, A. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the 15th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Paris, France* (2009), pp. 119–128.

[29] Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., and Züfle, A. Probabilistic frequent pattern growth for itemset mining in uncertain databases. In *Scientific and Statistical Database Management*. 2012, pp. 38–55.

[30] Bernecker, T., Kriegel, H.-P., Renz, M., and Züfle, A. Hot item detection in uncertain data. In *Proceedings of the 13rd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Bangkok, Thailand* (2009).

[31] Beskales, G., Soliman, M., and Ilyas, I. Efficient search for the top-k probable nearest neighbors in uncertain databases. *PVLDB 1* (2008), 326–339.

[32] Bloch, I. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition 32*, 11 (1999), 1873–1895.

[33] Bohannon, P., Fan, W., Flaster, M., and Rastogi, R. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Baltimore, ML* (2005), pp. 143–154.

[34] Böhm, C., Braunmüller, B., Breunig, M. M., and Kriegel, H.-P. "High Performance Clustering Based on the Similarity Join.". In *Proceedings of the 9th International Conference on Information and Knowledge Management (CIKM), Washington, D.C.* (2000), pp. 298–305.

[35] Böhm, C., Pryakhin, A., and Schubert, M. The Gauss-tree: Efficient object identification of probabilistic feature vectors. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, GA* (2006), p. 9.

[36] Böhm, C., Pryakhin, A., and Schubert, M. Probabilistic ranking queries on gaussians. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM), Vienna, Austria* (2006), pp. 169–178.

[37] Borzsony, S., Kossmann, D., and Stocker, K. The skyline operator. In *Data Engineering, 2001. Proceedings. 17th International Conference on* (2001), pp. 421 –430.

[38] Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. Lof: Identifying density-based local outliers. In *SIGMOD Conference* (2000), pp. 93–104.

[39] Brinkhoff, T., Kriegel, H.-P., and Seeger, B. Efficient processing of spatial joins using R-trees. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Washington, D.C.* (1993), pp. 237–246.

[40] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[41] Cam, L. L. An approximation theorem for the Poisson binomial distribution. In *Pacific Journal of Mathematics* (1960), vol. 10.

[42] Chakrabarti, D., and Faloutsos, C. Graph mining: Laws, generators, and algorithms. In *ACM Comput. Surv., 38(1), New York, NY, USA* (2006).

[43] Chapman, G., Cleese, J., Gilliam, T., Idle, E., Jones, T., and Palin, M. Monty python's the life of brian, 1979.

[44] Cheema, M. A., Lin, X., Wang, W., Zhang, W., and Pei, J. Probabilistic reverse nearest neighbor queries on uncertain data. *IEEE Trans. Knowl. Data Eng. 22*, 4 (2010), 550–564.

[45] Chen, J., and Cheng, R. Efficient evaluation of imprecise location-dependent queries. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey* (2007), pp. 586–595.

[46] Chen, L., and Lian, X. Dynamic skyline queries in metric spaces. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT), Nantes, France* (2008), pp. 333–343.

[47] Chen, Z., Shen, H. T., and Zhou, X. Discovering popular routes from trajectories. In *Proceedings of the 27th International Conference on Data Engineering (ICDE), Hannover, Germany* (2011), pp. 900–911.

[48] Cheng, R., Chen, J., Mokbel, M. F., and Chow, C.-Y. Probabilistic verifiers: Evaluating constrained nearest-neighbor queries over uncertain data. In *Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico* (2008), pp. 973–982.

[49] Cheng, R., Chen, L., Chen, J., and Xie, X. Evaluating probability threshold k-nearest-neighbor queries over uncertain data. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT), Saint-Petersburg, Russia* (2009), pp. 672–683.

[50] Cheng, R., Kalashnikov, D. V., and Prabhakar, S. Evaluating probabilistic queries over imprecise data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Diego, CA* (2003), pp. 551–562.

[51] Cheng, R., Kalashnikov, D. V., and Prabhakar, S. Querying imprecise data in moving object environments. *IEEE Trans. Knowl. Data Eng. 16*, 9 (2004), 1112–1127.

[52] Cheng, R., Xia, Y., Prabhakar, S., Shah, R., and Vitter, J. Efficient indexing methods for probabilistic threshold queries over uncertain data. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada* (2004), pp. 876–887.

[53] Chui, C. K., and Kao, B. A decremental approach for mining frequent itemsets from uncertain data. In *Proceedings of the 12nd Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Osaka, Japan* (2008), pp. 64–75.

[54] Chui, C.-K., Kao, B., and Hung, E. Mining frequent itemsets from uncertain data. In *In Proceedings of the 11th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-2007), Nanjing, China, May 22-25* (2007), pp. 47–58.

[55] Cormode, G., Li, F., and Yi, K. Semantics of ranking queries for probabilistic data and expected results. In *Proceedings of the 25th International Conference on Data Engineering (ICDE), Shanghai, China* (2009), pp. 305–316.

[56] Dalvi, N. N., Ré, C., and Suciu, D. Probabilistic databases: diamonds in the dirt. *Commun. ACM 52*, 7 (2009), 86–94.

[57] Dalvi, N. N., and Suciu, D. Efficient query evaluation on probabilistic databases. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada* (2004), pp. 864–875.

[58] Delling, D., Sanders, P., Schultes, D., and Wagner, D. Highway hierarchies star. In *In Proc. 9th DIMACS Implementation Challenge* (2006).

[59] Deshpande, A., Guestrin, C., Madden, S., Hellerstein, J. M., and Hong, W. Model-driven data acquisition in sensor networks. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada* (2004), pp. 588–599.

[60] Dijkstra, E. W. A note on two problems in connection with graphs. *Numerische Mathematik 1* (1959), 269–271.

[61] Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., Xu, N., and Züfle, A. Reverse k-nearest neighbor monitoring on mobile objects. In *Proc. ACM 17th International Workshop on Advances in Geographic Information Systems (ACM GIS), San Jose, CA* (2010).

[62] Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., Xu, N., and Züfle, A. Reverse k-nearest neighbor monitoring on mobile objects. In *GIS* (2010), pp. 494–497.

[63] Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., and Züfle, A. Constrained reverse nearest neighbor search on mobile objects. In *Proc. ACM 16th International Workshop on Advances in Geographic Information Systems (ACM GIS), Seattle, WA* (2009), pp. 197–206.

[64] Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., and Züfle, A. Incremental reverse nearest neighbor ranking in vector spaces. In *SSTD* (2009), pp. 265–282.

[65] Emrich, T., Kriegel, H.-P., Kröger, P., Renz, M., and Züfle, A. Boosting spatial pruning: On optimal pruning of MBRs. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Indianapolis, IN* (2010), pp. 39–50.

[66] Emrich, T., Kriegel, H.-P., Mamoulis, N., Renz, M., and Züfle, A. Indexing uncertain spatio-temporal data. In *Proceedings of the 21st International Conference on Information and Knowledge Management (CIKM), Hawaii* (2012), pp. 395–404.

[67] Esseen, C.-G. *On the Liapunoff limit of error in the theory of probability.* Arkiv för matematik, astronomi och fysik, 1942.

[68] Ester, M., Kriegel, H.-P., Sander, J., and Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd ACM International Conference on Knowledge Discovery and Data Mining (KDD), Portland, OR* (1996), pp. 226–231.

[69] Feller, W. *The Fundamental Limit Theorems in Probability.* Bull Amer. Math. Soc., 1945.

[70] Feller, W. *The Strong Law of Large Numbers, in An Introduction to Probability Theory and Its Applications.* New York: Wiley, 1968.

[71] Follmann, A., Nascimento, M. A., Züfle, A., Renz, M., Kröger, P., and Kriegel, H.-P. Continuous probabilistic count queries in wireless sensor networks. In *Proceedings of the 12th International Symposium on Spatial and Temporal Databases (SSTD), Minneapolis, MN* (2011), pp. 279–296.

[72] Frentzos, E., Gratsias, K., Pelekis, N., and Theodoridis, Y. Algorithms for nearest neighbor search on moving object trajectories. *Geoinformatica 11*, 2 (2007), 159–193.

[73] Fu, L., Sun, D., and Rilett, L. R. Heuristic shortest path algorithms for transportation applications: state of the art. In *Computers in Operations Research, 33(11):3324-3343* (2006).

[74] Fuhr, N., and Rölleke, T. A probabilistic relational algebra for the integration of information retrieval and database systems. *ACM Trans. Inf. Syst. 15*, 1 (1997), 32–66.

[75] Gaede, V., and Gnther, O. Multidimensional access methods. *ACM Computing Surveys 30*, 2 (1998), 170–231.

[76] Gaffney, S., and Smyth, P. Trajectory clustering with mixtures of regression models. In *Proceedings of the 5th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, CA*, pp. 63–72.

[77] Geurts, K., Wets, G., Brijs, T., and Vanhoof, K. Profiling high frequency accident locations using association rules. In *Proceedings of the 82nd Annual Transportation Research Board, Washington DC. (USA), January 12-16* (2003), p. 18pp.

[78] Gonzalez, H., Han, J., Li, X., Myslinska, M., and Sondag, J. P. Adaptive fastest path computation on a road network: A traffic mining approach. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), Vienna, Austria* (2007), pp. 794–805.

[79] Güting, R. H. An introduction to spatial database systems. *VLDB Journal 3*, 4 (1994), 357–399.

[80] Güting, R. H., Behr, T., and Xu, J. Efficient k-nearest neighbor search on moving object trajectories. *VLDB J. 19*, 5 (2010), 687–714.

[81] Güting, R. H., and Schneider, M. *Moving Objects Databases*. Morgan Kaufmann, 2005.

[82] Guttman, A. R-Trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Boston, MA* (1984), pp. 47–57.

[83] Hadjieleftheriou, M., Kollios, G., Gunopulos, D., and Tsotras, V. Online discovery of dense areas in spatio-temporal databases. In *Proceedings of the 8th International Symposium on Spatial and Temporal Databases (SSTD), Santorini Island, Greece* (2003), pp. 306–324.

[84] Hadjieleftheriou, M., Kollios, G., Tsotras, J., and Gunopulos, D. Indexing spatiotemporal archives. *The VLDB Journal 15*, 2 (2006), 143–164.

[85] Han, J., Pei, J., and Yin, Y. Mining frequent patterns without candidate generation. *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX 29* (2000), 1–12.

[86] Hettich, S., and Bay, S. D. The uci kdd archive., 1999.

[87] Hey, T., Tansley, S., , and Tolle, K. *The Fourth Paradigm. Data- Intensive Scientific Discovery, Microsoft Research.* 2009.

[88] Hjaltason, G. R., and Samet, H. Ranking in spatial databases. In *Proceedings of the 4th International Symposium on Large Spatial Databases (SSD), Portland, ME* (1995), pp. 83–95.

[89] Hodges, J., and le Cam, L. *The Poisson Approximation to the Poisson Binomial Distribution. The Annals of Mathematical Statistics.* Institute of Mathematical Statistics, 1959.

[90] Hoeffding, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58 (301)*, 13–30.

[91] Hua, M., Pei, J., Zhang, W., and Lin, X. Ranking queries on uncertain data: a probabilistic threshold approach. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008* (2008), pp. 673–686.

[92] Huang, Y., Pei, J., and Xiong, H. Mining co-location patterns with rare events from spatial data sets. In *Geoinformatica* (2006), vol. 10, pp. 239–260.

[93] Huang, Y., Shekhar, S., and Xiong, H. Discovering co-location patterns from spatial data sets: A general approach. In *IEEE Transactions on Knowledge and Data Engineering* (2004), vol. 16, pp. 1472–1485.

[94] Huang, Y.-K., Liao, S.-J., and Lee, C. Efficient continuous k-nearest neighbor query processing over moving objects with uncertain speed and direction. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management (SSDBM), Hong Kong, China* (2008), pp. 549–557.

[95] Iijima, Y., and Ishikawa, Y. Finding probabilistic nearest neighbors for query objects with imprecise locations. In *Proceedings of the 10th International Conference on Mobile Data Management (MDM), Taipei, Taiwan* (2009), pp. 52–61.

[96] Iwerks, G. S., Samet, H., and Smith, K. Continuous k-nearest neighbor queries for continuously moving points with updates. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), Berlin, Germany* (2003), VLDB Endowment, pp. 512–523.

[97] Jing, N., Huang, Y.-W., and Rundensteiner, E. A. Hierarchical optimization of optimal path finding for transportaton applications. In *Proceedings of the 5th International Conference on Information and Knowledge Management (CIKM), Rockville, MD* (1996), pp. 261–268.

[98] Jung, S., and Pramanik, S. Hiti graph model of topographical road maps in navigation systems. In *Proceedings of the 12th International Conference on Data Engineering (ICDE), New Orleans, LA* (1996), pp. 76–84.

[99] Kalnis, P., Mamoulis, N., and Bakiras, S. On discovering moving clusters in spatio-temporal data. In *Proceedings of the 9th International Symposium on Spatial and Temporal Databases (SSTD), Angra dos Reis, Brazil* (2005), pp. 364–381.

[100] Kanoulas, E., Du, Y., Xia, T., and Zhang, D. Finding fastest paths on a road network with speed patterns. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, GA* (2006), p. 10.

[101] Karlin, S., and Taylor, H. M. *A First Course in Stochastic Processes*, vol. 2. Academic Pr Inc, 1975.

[102] Keogh, E. Exact indexing of dynamic time warping. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China* (2002), pp. 406–417.

[103] Kollios, G., Gunopulos, D., and Tsotras, V. Nearest neighbor queries in a mobile environment. In *Spatio-Temporal Database Management* (1999), Springer, pp. 119–134.

[104] Korn, F., and Muthukrishnan, S. Influenced sets based on reverse nearest neighbor queries. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX* (2000), pp. 201–212.

[105] Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., and Protopapas, Z. Fast nearest neighbor search in medical image databases. In *Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB), Bombay, India* (1996), pp. 215–226.

[106] Kozawa, Y., Amagasa, T., and Kitagawa, H. Gpu acceleration of probabilistic frequent itemset mining from uncertain databases. In *Proceedings of the 21st ACM international conference on Information and knowledge management* (2012), pp. 892–901.

[107] Kriegel, H.-P., Kröger, P., Renz, M., Züfle, A., and Katzdobler, A. Incremental reverse nearest neighbor ranking. In *Proceedings of the 25th International Conference on Data Engineering (ICDE), Shanghai, China* (2009), pp. 1560–1567.

[108] Kriegel, H.-P., Kröger, P., Renz, M., Züfle, A., and Katzdobler, A. Reverse k-nearest neighbor search based on aggregate point access methods. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management (SSDBM), New Orleans, LA* (2009), pp. 444–460.

[109] Kriegel, H.-P., Kunath, P., Pfeifle, M., and Renz, M. Probabilistic similarity join on uncertain data. In *Proceedings of the 11th International Conference on Database Systems for Advanced Applications (DASFAA), Singapore* (2006), pp. 295–309.

[110] Kriegel, H.-P., Kunath, P., and Renz, M. Probabilistic nearest-neighbor query on uncertain objects. In *Proceedings of the 12th International Conference on Database Systems for Advanced Applications (DASFAA), Bangkok, Thailand* (2007), pp. 337–348.

[111] Kriegel, H.-P., and Pfeifle, M. Density-based clustering of uncertain data. In *Proceedings of the 11th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Chicago, IL* (2005), pp. 672–677.

[112] Kriegel, H.-P., and Pfeifle, M. Efficient and effective server-sided distributed clustering. In *Proceedings of the 14th International Conference on Information and Knowledge Management (CIKM), Bremen, Germany* (2005).

[113] Kriegel, H.-P., Renz, M., Schubert, M., and Züfle, A. Statistical density prediction in traffic networks. In *Proceedings of the 8th SIAM International Conference on Data Mining (SDM), Atlanta, GA* (2008), SIAM, pp. 692–703.

[114] Kriegel, H.-P., Renz, M., Schubert, M., and Züfle, A. Efficient traffic density prediction in road networks using suffix trees. *KI Journal 26*, 3 (2012), 233–240.

[115] Kriegel, H.-P., Schmidt, T., and Seidl, T. 3d similarity search by shape approximation. In *Proceedings of the 5th International Symposium on Large Spatial Databases (SSD), Berlin, Germany* (1997), pp. 11–28.

[116] Kuijpers, B., and Othman, W. Trajectory databases: Data models, uncertainty and complete query languages. In *JCSS* (2010), pp. 538–560.

[117] Lange, K. Numerical analysis for statisticians. In *Statistics and computing* (1999).

[118] Lazaridis, I., and Mehrotra, S. Progressive approximate aggregate queries with a multi-resolution tree structure. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Santa Barbara, CA* (2001), pp. 401–412.

[119] Lee, J., Han, J., and Whang, K. Trajectory clustering: A partition-and-group framework. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Beijing, China* (2007), pp. 593–604.

[120] Lehning, M., Dawes, N., Bavay, M., Parlange, M., Nath, S., and Zhao, F. Instrumenting the Earth: Next-Generation Sensor Networks and Environmental Science. *The Fourth Paradigm. Data-Intensive Scientific Discovery* (2011), 45–51.

[121] Leung, C. K.-S., Carmichael, C. L., and Hao, B. Efficient mining of frequent patterns from uncertain data. In *ICDMW '07: Proceedings of the Seventh IEEE International Conference on Data Mining Workshops* (2007), pp. 489–494.

[122] Li, G., Li, Y., Shu, L., and Fan, P. Cknn query processing over moving objects with uncertain speeds in road networks. In *APWeb* (2011), pp. 65–76.

[123] Li, J., and Deshpande, A. Consensus answers for queries over probabilistic databases. In *Symposium on Principles of Database Systems (PODS), Providence, Rhode Island.* (2009), pp. 259–268.

[124] Li, J., and Deshpande, A. Ranking continuous probabilistic datasets. *Proceedings of the 36nd International Conference on Very Large Data Bases (VLDB), Singapore 3*, 1 (2010), 638–649.

[125] Li, J., Saha, B., and Deshpande, A. A unified approach to ranking in probabilistic databases. *Proceedings of the 35nd International Conference on Very Large Data Bases (VLDB), Lyon, France 2*, 1 (2009), 502–513.

[126] Li, J., Saha, B., and Deshpande, A. A unified approach to ranking in probabilistic databases. *VLDB Journal 20*, 2 (2011), 249–275.

[127] Li, J., Zou, Z., and Gao, H. Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. *The VLDB Journal 21* (2012), 753–777.

[128] Li, X., Han, J., Lee, J.-G., and Gonzalez, H. Traffic density-based discovery of hot routes in road networks. In *Proceedings of the 10th International Symposium on Spatial and Temporal Databases (SSTD), Boston, MA* (2007), pp. 441–459.

[129] Lian, X., and Chen, L. Probabilistic ranked queries in uncertain databases. In *Proceedings of the 12th International Conference on Extending Database Technology (EDBT), Nantes, France* (2008), pp. 511–522.

[130] Lian, X., and Chen, L. Efficient processing of probabilistic reverse nearest neighbor queries over uncertain data. *VLDB Journal 18*, 3 (2009), 787–808.

[131] Lian, X., and Chen, L. Probabilistic inverse ranking queries over uncertain data. In *Proceedings of the 14th International Conference on Database Systems for Advanced Applications (DASFAA), Brisbane, Australia* (2009), pp. 35–50.

[132] Ljosa, V., and Singh, A. Probabilistic segmentation and analysis of horizontal cells. In *Proceedings of the 6th International Conference on Data Mining (ICDM), Hong Kong, China* (2006), pp. 980 – 985.

[133] Ljosa, V., and Singh, A. K. Apla: Indexing arbitrary probability distributions. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey* (2007), pp. 946–955.

[134] Mamoulis, N. Co-location pattern. In *Encyclopedia of GIS*. 2008, p. 98.

[135] Mamoulis, N. Co-location patterns, algorithms. In *Encyclopedia of GIS*. 2008, pp. 103–107.

[136] Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., and Cheung, D. W. Mining, indexing, and querying historical spatiotemporal data. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA* (2004), pp. 236–245.

[137] Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., and Byers, A. H. Big data: The next frontier for innovation, competition, and productivity. *McKinsey & Company Report* (May 2009).

[138] Mason, O., and Verwoerd, M. Graph theory and networks in biology.

[139] Mokhtar, H., and Su, J. Universal trajectory queries for moving object databases. In *Mobile Data Management (MDM)* (2004), pp. 133–144.

[140] Nascimento, M. A., Renz, M., Emrich, T., Mouratidis, K., and Züfle, A. First international acm workshop on managing and mining contextually rich geo-spatial data., 2014. `www.dbs.ifi.lmu.de/georich14/index.php`.

[141] Niedermayer, J., Züfle, A., Emrich, T., Renz, M., Mamoulis, N., Chen, L., and Kriegel, H.-P. Probabilistic nearest neighbor queries on uncertain moving object trajectories. In *To Appear in the 40th International Conference on Very Large Databases (VLDB)* (2014).

[142] Oswald, R. K., Scherer, W. T., and Smith, B. L. Traffic flow forecasting using approximate nearest neighbor nonparametric regression. In *Final project of ITS Center project: Traffic forecasting: non-parametric regressions, December* (2000).

[143] Pallottino, S., and Scutella, M. G. Shortest path algorithms in transportation models: classical and innovative aspects. In *Technical Report TR-97-06, 14* (1997).

[144] Papadias, D., Kalnis, P., Zhang, J., and Tao, Y. Efficient olap operations in spatial data warehouses. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD), Redondo Beach, CA* (2001), pp. 443–459.

[145] Papoulis, A. *Probability, Random Variables, and Stochastic Processes, 2nd ed.* McGraw-Hill, 1984.

[146] Patroumpas, K., Papamichalis, M., and Sellis, T. K. Probabilistic range monitoring of streaming uncertain positions in geosocial networks. In *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management (SSDBM), Crete, Greece* (2012), pp. 20–37.

[147] Pei, J., Hua, M., Tao, Y., and Lin, X. Query answering techniques on uncertain and probabilistic data: tutorial summary. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Vancouver, BC* (2008), pp. 1357–1364.

[148] Pei, J., Jiang, B., Lin, X., and Yuan, Y. Probabilistic skylines on uncertain data. In *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB), Vienna, Austria* (2007), pp. 15–26.

[149] Peng, S., Urbanc, B., Cruz, L., Hyman, B. T., and Stanley, H. E. Neuron recognition by parallel potts segmentation. *Proceedings of the National Academy of Sciences of the United States of America 100*, 7 (2003), 3847–3852.

[150] Pfoser, D., and Jensen, C. S. Capturing the uncertainty of moving-object representations. In *Proceedings of the 6th International Symposium on Large Spatial Databases (SSD), Hong-Kong* (1999), pp. 111–132.

[151] Pfoser, D., Jensen, C. S., and Theodoridis, Y. Novel approaches to the indexing of moving object trajectories. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB), Cairo, Egypt* (2000), pp. 396–406.

[152] Poisson, S. *Recherches sur la probilit des Jugements.* 1837.

[153] Prasad Sistla, A., Wolfson, O., Chamberlain, S., and Dao, S. Modeling and querying moving objects. In *Proceedings of the 13th International Conference on Data Engineering (ICDE), Birmingham, U.K.* (1997), IEEE, pp. 422–432.

[154] R, C., Letchner, J., Balazinksa, M., and Suciu., D. Event queries on correlated probabilistic streams. *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Vancouver, BC* (2008), 715–728.

[155] Re, C., Dalvi, N., and Suciu, D. "Efficient top-k query evaluation on probalistic databases". In *Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey* (2007), pp. 886–895.

[156] Re, C., Dalvi, N. N., and Suciu, D. Query evaluation on probabilistic databases. *IEEE Data Eng. Bull. 29*, 1 (2006), 25–31.

[157] Renz, M., Cheng, R., Kriegel, H.-P., Züfle, A., and Bernecker, T. Similarity search and mining in uncertain databases. *Proceedings of the 36nd International Conference on Very Large Data Bases (VLDB), Singapore 3*, 2 (2010), 1653–1654.

[158] Renze, J., and Weisstein, E. Law of large numbers.

[159] Roussopoulos, N., Kelley, S., and Vincent, F. Nearest neighbor queries. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), San Jose, CA* (1995), pp. 71–79.

[160] Saltenis, S., Jensen, C. S., Leutenegger, S. T., and Lopez, M. A. Indexing the positions of continuously moving objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Dallas, TX* (2000), pp. 331–342.

[161] Sander, J., Ester, M., Kriegel, H.-P., and Xu, X. Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications. *Data Mining and Knowledge Discovery 2*, 2 (1998), 169–194.

[162] Sanders, P., and Schultes, D. Highway hierarchies hasten exact shortest path queries. In *In Proc. 17th European Symposium on Algorithms (ESA)* (2005), pp. 568–579.

[163] Sarma, A. D., Benjelloun, O., Halevy, A. Y., and Widom, J. Working models for uncertain data. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, GA* (2006), p. 7.

[164] Schneider, M. Uncertainty management for spatial datain databases: Fuzzy spatial data types. In *Advances in Spatial Databases*, vol. 1651. 1999, pp. 330–351.

[165] Schneider, M. Fuzzy topological predicates, their properties, and their integration into query languages. In *Proc. ACM 8th International Workshop on Advances in Geographic Information Systems (ACM GIS), Atlanta, GA* (2001), pp. 9–14.

[166] Sen, P., and Deshpande, A. Representing and querying correlated tuples in probabilistic databases. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey* (2007), pp. 596–605.

[167] Shekhar, S., and Huang, Y. Co-location rules mining: A summary of results. In *Proceedings of the 7th International Symposium on Spatial and Temporal Databases (SSTD), Redondo Beach, CA* (2001), pp. 236–256.

[168] Shekhar, S., Lu, C.-T., Chawla, S., and Zhang, P. Data mining and visualization of twin-cities traffic data. In *Technical Report TR 01-015, Dept. of CSE, Univ. of Minnesota* (2000).

[169] Shevtsova, I. G. An improvement of convergence rate estimates in the lyapunov theorem. In *Doklady Mathematics Vol. 82* (2010), pp. 862–864.

[170] Singh, S., Mayfield, C., Mittal, S., Prabhakar, S., Hambrusch, S. E., and Shah, R. Orion 2.0: native support for uncertain data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Vancouver, BC* (2008), pp. 1239–1242.

[171] Soliman, M., and Ilyas, I. Ranking with uncertain scores. In *Proceedings of the 25th International Conference on Data Engineering (ICDE), Shanghai, China* (2009), pp. 317–328.

[172] Soliman, M. A., Ilyas, I. F., and Chang, K. C.-C. Top-k query processing in uncertain databases. In *Proceedings of the 23rd International Conference on Data Engineering (ICDE), Istanbul, Turkey* (2007), pp. 896–905.

[173] Stanoi, I., Agrawal, D., and Abbadi, A. E. Reverse nearest neighbor queries for dynamic databases. In *Proceedings of the ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery (DMKD), Dallas, TX* (2000), pp. 44–53.

[174] Stein, C. Approximate Computation of Expectations. *Institute of Mathematical Statistics Lecture Notes - Monograph Series 7* (1986).

[175] Tang, J., Chen, Z., Fu, A. W.-C., and Cheung, D. W. Enhancing effectiveness of outlier detections for low density patterns. In *Proceedings of the 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD), Taipei, Taiwan* (2002), pp. 535–548.

[176] Tang, X., and Kainz, W. Analysis of topological relations between fuzzy regions in a general fuzzy topological space. In *Symposium on Geospatial Theory, Processing and Applications* (2002).

[177] Tao, Y., Cheng, R., Xiao, X., Ngai, W. K., Kao, B., and Prabhakar, S. Indexing multi-dimensional uncertain data with arbitrary probability density functions. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), Trondheim, Norway* (2005), pp. 922–933.

[178] Tao, Y., Faloutsos, C., Papadias, D., and Liu, B. Prediction and indexing of moving objects with unknown motion patterns. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Paris, France* (2004), pp. 611–622.

[179] Tao, Y., and Papadias, D. Time-parameterized queries in spatio-temporal databases. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Madison, WI* (2002), pp. 334–345.

[180] Tao, Y., Papadias, D., and Lian, X. Reverse kNN search in arbitrary dimensionality. In *Proceedings of the 30th International Conference on Very Large Data Bases (VLDB), Toronto, Canada* (2004), pp. 744–755.

[181] Tao, Y., Papadias, D., and Shen, Q. Continuous nearest neighbor search. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB), Hong Kong, China* (2002), pp. 287–298.

[182] Tao, Y., Papadias, D., and Shen, Q. Continuous nearest neighbor search. In *VLDB* (2002), pp. 287–298.

[183] Tao, Y., Papadias, D., and Sun, J. The tpr*tree: An optimized spatio-temporal access method for predictive queries. In *Proceedings of the 29th International Conference on Very Large Data Bases (VLDB), Berlin, Germany* (2003), pp. 790–801.

[184] Tao, Y., Xiao, X., and Cheng, R. Range search on multidimensional uncertain data. *ACM Trans. Database Syst. 32*, 3 (Aug. 2007).

[185] Tao, Y., Yiu, M. L., and Mamoulis, N. Reverse nearest neighbor search in metric spaces. *IEEE Trans. Knowl. Data Eng. 18*, 9 (2006), 1239–1252.

[186] Tong, H., and Faloutsos, C. Center-piece subgraphs: problem definition and fast solutions. In *Proceedings of the 12nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA* (2006), pp. 404–413.

[187] Tong, Y., Chen, L., Cheng, Y., and Yu, P. S. Mining frequent itemsets over uncertain databases. *Proceedings of the 38th International Conference on Very Large Data Bases (VLDB), Istanbul, Turkey 5*, 11 (2012), 1650–1661.

[188] Trajcevski, G., Choudhary, A. N., Wolfson, O., Ye, L., and Li, G. Uncertain range queries for necklaces. In *11th International Conference on Mobile Data Management (MDM 2010), Kansas City, Missouri* (2010), pp. 199–208.

[189] Trajcevski, G., Tamassia, R., Cruz, I. F., Scheuermann, P., Hartglass, D., and Zamierowski, C. Ranking continuous nearest neighbors for uncertain trajectories. *VLDB J. 20*, 5 (2011), 767–791.

[190] Trajcevski, G., Tamassia, R., Ding, H., Scheuermann, P., and Cruz, I. F. Continuous probabilistic nearest-neighbor queries for uncertain trajectories. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT), Saint-Petersburg, Russia* (2009), pp. 874–885.

[191] Trajcevski, G., Wolfson, O., Hinrichs, K., and Chamberlain, S. Managing uncertainty in moving objects databases. *ACM Trans. Database Syst. 29*, 3 (2004), 463–507.

[192] Trajcevski, G., Wolfson, O., Zhang, F., and Chamberlain, S. The geometry of uncertainty in moving objects databases. In *Proceedings of the 8th International Conference on Extending Database Technology (EDBT), Prague, Czech Republic* (2002), pp. 233–250.

[193] Tran, T. T., Peng, L., Li, B., Diao, Y., and Liu, A. Pods: a new model and processing algorithms for uncertain data streams. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Indianapolis, IN* (2010), pp. 159–170.

[194] Valiant, L. The complexity of enumeration and reliability problems. In *SIAM Journal of Computing* (1979), pp. 410–421.

[195] Wang, L., Cheng, R., Lee, S. D., and Cheung, D. Accelerating probabilistic frequent itemset mining: a model-based approach. In *Proceedings of the 19th International Conference on Information and Knowledge Management (CIKM), Toronto, ON, Canada* (2010), pp. 429–438.

[196] Wang, L., Cheung, D.-L., Cheng, R., Lee, S.-D., and Yang, X. Efficient mining of frequent item sets on large uncertain databases. *Knowledge and Data Engineering, IEEE Transactions on 24*, 12 (2012), 2170–2183.

[197] Wang, L., Wu, P., and Chen, H. Finding probabilistic prevalent colocations in spatially uncertain data sets. *IEEE Transactions on Knowledge and Data Engineering 25*, 4 (2013), 790–804.

[198] Wijsen, J. Database repairing using updates. *ACM Trans. Database Syst. 30*, 3 (2005).

[199] Xu, C., Gu, Y., Chen, L., Qiao, J., and Yu, G. Interval reverse nearest neighbor queries on uncertain data with markov correlations. In *Proceedings of the 29th International Conference on Data Engineering (ICDE), Brisbane, Australia* (2013).

[200] Yan Huang, L. Z., and Zhang, P. Finding sequential patterns from a massive number of spatio-temporal events. In *Proceedings of the 6th SIAM International Conference on Data Mining (SDM), Bethesda, MD* (2006).

[201] Yeh, M.-Y., Wu, K.-L., Yu, P. S., and Chen, M. Proud: A probabilistic approach to processing similarity queries over uncertain data streams. In *Proceedings of the 13th International Conference on Extending Database Technology (EDBT), Saint-Petersburg, Russia* (2009), pp. 684–695.

[202] Yi, K., Li, F., Kollios, G., and Srivastava, D. Efficient processing of top-k queries in uncertain databases. In *Proceedings of the 24th International Conference on Data Engineering (ICDE), Cancun, Mexico* (2008), pp. 1406–1408.

[203] Yi, K., Li, F., Kollios, G., and Srivastava, D. Efficient processing of top-k queries in uncertain databases with x-relations. *IEEE Trans. Knowl. Data Eng. 20*, 12 (2008), 1669–1682.

[204] Yiu, M. L., and Mamoulis, N. Reverse nearest neighbors search in ad-hoc subspaces. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE), Atlanta, GA* (2006), p. 76.

[205] Yiu, M. L., Mamoulis, N., Dai, X., Tao, Y., and Vaitis, M. Efficient evaluation of probabilistic advanced spatial queries on existentially uncertain data. *Knowledge and Data Engineering, IEEE Transactions on 21*, 1 (Jan.), 108–122.

[206] Yiu, M. L., Papadias, D., Mamoulis, N., and Tao, Y. Reverse nearest neighbors in large graphs. *IEEE Trans. Knowl. Data Eng. 18*, 4 (2006), 540–553.

[207] Yu Zheng, X. Z. *Moving Objects Databases. Foreword by Jiawei Han. Editorial Board: Ralf Hartmut Güting, Hans-Peter Kriegel, Hanan Samet.* Springer, 2011.

[208] Yuan, J., Zheng, Y., Xie, X., and Sun, G. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), San Diego, C.A.* (2011), pp. 316–324.

[209] Yuan, J., Zheng, Y., Zhang, C., Xie, W., Xie, X., and Huang, Y. T-drive: Driving directions based on taxi trajectories. In *Proc. ACM GIS* (2010).

[210] Zacharias, N., and Zacharias, M. I. The twin astrographic catalog on the hipparcos system. *The Astronomical Journal 118*, 5 (1999), 2503–2510.

[211] Zadeh, L. A. Fuzzy sets. *Information and Control 8*, 3 (1965), 338–353.

[212] Zhang, Q., Li, F., and Yi, K. Finding frequent items in probabilistic data. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Vancouver, BC* (2008), pp. 819–832.

[213] Zhang, X., and Chomicki, J. On the semantics and evaluation of top-k queries in probabilistic databases. In *Proceedings of the 24th International Conference on Data Engineering Workshops, ICDE 2008, April 7-12, 2008, Cancún, México* (2008), pp. 556–563.

[214] Zhang, X., Mamoulis, N., Cheung, D. W., and Shou, Y. Fast mining of spatial collocations. In *Proceedings of the 10th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Seattle, WA* (2004), pp. 384–393.

[215] Zheng, K., Fung, G. P. C., and Zhou, X. K-nearest neighbor search for fuzzy objects. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD), Indianapolis, IN* (2010), pp. 699–710.

[216] Zheng, Y., and Zhou, X. *Computing with Spatial Trajectories*. Springer, 2011.

[217] Zimányi, E. Query evaluation in probabilistic relational databases. *Theor. Comput. Sci. 171*, 1-2 (1997), 179–219.

[218] Zou, Z., Gao, H., and Li, J. Discovering frequent subgraphs over uncertain graph databases under probabilistic semantics. In *Proceedings of the 16th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Washington, D.C.* (2010), pp. 633–642.

[219] Zou, Z., Li, J., Gao, H., and Zhang, S. Mining frequent subgraph patterns from uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on 22*, 9 (2010), 1203–1218.

[220] Zwillinger, D., and Kokoska, S. *CRC Standard Probability and Statistics Tables and Formulae*. CRC Press, 2000.