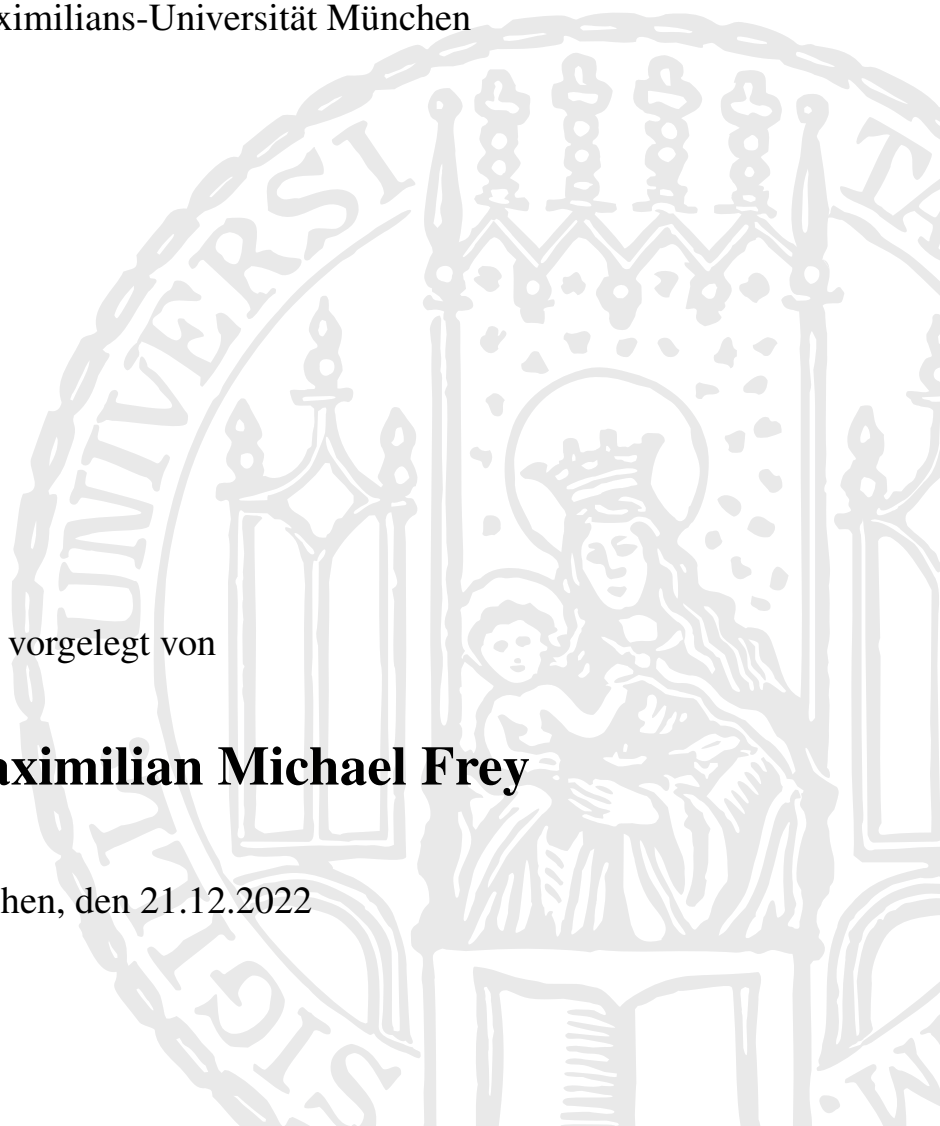
Learning from Complex Networks

Dissertation

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



vorgelegt von

Christian Maximilian Michael Frey

München, den 21.12.2022

Tag der Einreichung: 21.12.2022

Erstgutachter: Apl. Prof. Dr. Matthias Schubert
Ludwig-Maximilians-Universität München
Institut für Informatik
Lehrstuhl für Datenbanksysteme und Data Mining

Zweitgutachter: Prof. Dr. Achim Rettinger
Universität Trier
Computational Linguistics
Knowledge Representation Learning

Vorsitz: Prof. Dr. Albrecht Schmidt
Ludwig-Maximilians-Universität München
Institut für Informatik
Human-Centered Ubiquitous Media

Tag der Disputation: 19.06.2023

Eidesstattliche Versicherung

(siehe Promotionsordnung vom 12.07.2011, § 8, Abs. 2 Pkt. 5)

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

München, den 21.12.2022

Christian Maximilian Michael Frey

In Gedenken an meine Eltern.

'Wahr sind nur die Erinnerungen, die wir mit uns tragen;
die Träume, die wir spinnen, und die Sehnsüchte, die uns treiben.
Damit wollen wir uns bescheiden.'
- Die Feuerzangenbowle

Contents

Abstract

Graph Theory has proven to be a universal language for describing modern complex systems. The elegant theoretical framework of graphs drew the researchers' attention over decades. Therefore, graphs have emerged as a ubiquitous data structure in various applications where a relational characteristic is evident. Graph-driven applications are found, e.g., in social network analysis, telecommunication networks, logistic processes, recommendation systems, modeling kinetic interactions in protein networks, or the 'Internet of Things' (IoT) where modeling billions of interconnected web-enabled devices is of paramount importance.

This thesis dives deep into the challenges of modern graph applications. It proposes a robustified and accelerated spectral clustering model in homogeneous graphs and novel transformer-driven graph shell models for attributed graphs. A new data structure is introduced for probabilistic graphs to compute the information flow efficiently. Moreover, a metaheuristic algorithm is designed to find a good solution to an optimization problem composed of an extended vehicle routing problem. The thesis closes with an analysis of trend flows in social media data.

Detecting communities within a graph is a fundamental data mining task of interest in virtually all areas and also serves as an unsupervised preprocessing step for many downstream tasks. One most the most well-established clustering methods is *Spectral Clustering*. However, standard spectral clustering is highly sensitive to noisy input data, and the eigendecomposition has a high, cubic runtime complexity $O(n^3)$. Tackling one of these problems often exacerbates the other. This thesis presents a new model which accelerates the eigendecomposition step by replacing it with a Nyström approximation. Robustness is achieved by iteratively separating the data into a cleansed and noisy part of the data. In this process, representing the input data as a graph is vital to identify parts of the data being well connected by analyzing the vertices' distances in the eigenspace.

With the advances in deep learning architectures, we also observe a surge in research on graph representation learning. The *message-passing paradigm* in *Graph Neural Networks* (GNNs) formalizes a predominant heuristic for multi-relational and attributed graph data to learn node representations. In downstream applications, we can use the representations to tackle theoretical problems known as *node classification*, *graph classification/regression*, and *relation prediction*. However, a common issue in GNNs is known as *over-smoothing*. By increasing the number of iterations within the message-passing, the nodes' representations of the input graph align and

become indiscernible. This thesis shows an efficient way of relaxing the GNN architecture by employing a routing heuristic in the general workflow. Specifically, an additional layer routes the nodes' representations to dedicated experts. Each expert calculates the representations according to their respective GNN workflow. The definitions of distinguishable GNNs result from k-localized views starting from a central node. This procedure is referred to as *Graph **Sh**ell **A**ttention* (SEA), where experts process different subgraphs in a transformer-motivated fashion.

Reliable propagation of information through large communication networks, social networks, or sensor networks is relevant to applications concerning marketing, social analysis, or monitoring physical or environmental conditions. However, social ties of friendship may be obsolete, and communication links may fail, inducing the notion of uncertainty in such networks. This thesis addresses the problem of optimizing information propagation in uncertain networks given a constrained budget of edges. A specialized data structure, called *F-tree*, addresses two NP-hard subproblems: the computation of the expected information flow and the optimal choice of edges. The *F-tree* identifies independent components of a probabilistic input graph for which the information flow can either be computed analytically and efficiently or for which traditional Monte-Carlo sampling can be applied independently of the remaining network.

The next part of the thesis covers a graph problem from the *Operations Research* point of view. A new variant of the well-known vehicle routing problem (VRP) is introduced, where customers are served within a specific time window (TW), as well as flexible delivery locations (FL) including capacity constraints. The latter implies that each customer is scheduled in one out of a set of capacitated delivery service locations. Practically, the VRPTW-FL problem is relevant for applications in parcel delivery, routing with limited parking space, or, for example, in the scope of hospital-wide scheduling of physical therapists. This thesis presents a metaheuristic built upon a hybrid *Adaptive Large Neighborhood Search* (ALNS). Moreover, a backtracking mechanism in the construction phase is introduced to alter unsatisfactory decisions at early stages. In the computational study, hospital data is used to evaluate the utility of flexible delivery locations and various cost functions.

In the last part of the thesis, social media trends are analyzed, which yields insights into user sentiment and newsworthy topics. Such trends consist of bursts of messages concerning a particular topic within a time frame, significantly deviating from the average appearance frequency of the same subject. This thesis presents a method to classify trend archetypes to predict future dissemination by investigating the dissemination of such trends in space and time.

Generally, with the ever-increasing scale and complexity of graph-structured datasets and artificial intelligence advances, AI-backed models will inevitably play an important role in analyzing, modeling, and enhancing knowledge extraction from graph data.

Zusammenfassung

Die Graphentheorie hat sich zur einer universellen Sprache entwickelt, mit Hilfe derer sich moderne und komplexe Systeme und Zusammenhänge beschreiben lassen. Diese theoretisch elegante und gut fundierte Rahmenstruktur attrahierte über Dekaden hinweg die Aufmerksamkeit von Wissenschaftlern/-innen. In der heutigen Informationstechnologie-Landschaft haben sich Graphen längst zu einer allgegenwärtigen Datenstruktur in Anwendungen etabliert, innerhalb derer charakteristische Zusammenhangskomponenten eine zentrale Rolle spielen. Anwendungen, die über Graphen unterstützt werden, finden sich u.a. in der Analyse von sozialen Netzwerken, Telekommunikationsnetwerken, logistische Prozessverwaltung, Analyse von Empfehlungsdiensten, in der Modellierung kinetischer Interaktionen von Proteinstrukturen, oder auch im "Internet der Dinge" (engl.: 'Internet Of Things' (IoT)), welches das Zusammenspiel von abermillionen web-unterstützte Endgeräte abbildet und eine prädominierende Rolle für große IT-Unternehmen spielt.

Diese Dissertation beleuchtet die Herausforderungen moderner Graphanwendungen. Im Bereich homogener Netzwerken wird ein beschleunigtes und robustes spektrales Clusteringverfahren, sowie ein Modell zur Untersuchung von Teilgraphen mittels Transformer-Architekturen für attribuierte Graphen vorgestellt. Auf wahrscheinlichkeitsbasierten homogenen Netzwerken wird eine neue Datenstruktur eingeführt, die es erlaubt einen effizienten Informationsfluss innerhalb eines Graphen zu berechnen. Darüber hinaus wird ein Optimierungsproblem in Transportnetzwerken beleuchtet, sowie eine Untersuchung von Trendflüssen in sozialen Medien diskutiert.

Die Untersuchung von Verbünden (engl.: 'Clusters') von Graphdaten stellt einen Eckpfeiler im Bereich der Datengewinnung dar. Die Erkenntnisse sind nahezu in allen praktischen Bereichen von Relevanz und dient im Bereich des unüberwachten Lernens als Vorverarbeitungsschritt für viele nachgeschaltete Aufgaben. Einer der weit verbreitetsten Methodiken zur Verbundanalyse ist das spektrale Clustering. Die Qualität des spektralen Clusterings leidet, wenn die Eingabedaten sehr verrauscht sind und darüber hinaus ist die Eigenwertzerlegung mit $O(n^3)$ eine teure Operation und damit wesentlich für die hohe, kubische Laufzeitkomplexität verantwortlich. Die Optimierung von einem dieser Kriterien exazerbiert oftmals das verbleibende Kriterium. In dieser Dissertation wird ein neues Modell vorgestellt, innerhalb dessen die Eigenwertzerlegung über eine Nyström Annäherung beschleunigt wird. Die Robustheit wird über ein iteratives Verfahren erreicht, das die gesäuberten und die verrauschten Daten voneinander trennt. Die Darstel-

lung der Eingabedaten über einen Graphen spielt hierbei die zentrale Rolle, die es erlaubt die dicht verbundenen Teile des Graphen zu identifizieren. Dies wird über eine Analyse der Distanzen im Eigenraum erreicht.

Parallel zu neueren Erkenntnissen im Bereich des Deep Learnings lässt sich auch ein Forschungsdrang im repräsentativen Lernen von Graphen erkennen. Graph Neural Networks (GNN) sind eine neue Unterform von künstlich neuronalen Netzen (engl.: 'Artificial Neural Networks') auf der Basis von Graphen. Das Paradigma des sogenannten 'message-passing' in neuronalen Netzen, die auf Graphdaten appliziert werden, hat sich hierbei zur prädominierenden Heuristik entwickelt, um Vektordarstellungen von Knoten aus (multi-)relationalen, attribuierten Graphdaten zu lernen. Am Ende der Prozesskette können wir somit theoretische Probleme angehen und lösen, die sich mit Fragestellungen über die Klassifikation von Knoten oder Graphen, über regressive Ausdrucksmöglichkeiten bis hin zur Vorhersage von relationaler Verbindungen beschäftigen. Ein klassisches Problem innerhalb graphischer neuronaler Netze ist bekannt unter der Terminologie des 'over-smoothing' (dt.: 'Überglättens'). Es beschreibt, dass sich mit steigender Anzahl an Iterationen des wechselseitigen Informationsaustausches, die Knotenrepräsentationen im vektoriellen Raum angleichen und somit nicht mehr unterschieden werden können. In dieser Forschungsarbeit wird eine effiziente Methode vorgestellt, die die klassische GNN Architektur aufbricht und eine Vermittlerschicht in den herkömmlichen Verarbeitungsfluss einarbeitet. Konkret gesprochen werden hierbei Knotenrepräsentationen an ausgezeichnete Experten geschickt. Jeder Experte verarbeitet auf idiosynkratischer Basis die Knoteninformation. Ausgehend von einem Anfrageknoten liegt das Kriterium für die Unterscheidbarkeit von Experten in der restriktiven Verarbeitung lokaler Information. Diese neue Heuristik wird als 'Graph Shell Attention' (SEA) bezeichnet und beschreibt die Informationsverarbeitung unterschiedlicher Teilgraphen von Experten unter der Verwendung der Transformer-technologie.

Eine zuverlässige Weiterleitung von Informationen über größere Kommunikationsnetzwerken, sozialen Netzwerken oder Sensorennetzwerken spielen eine wichtige Rolle in Anwendungen der Marktanalyse, der Analyse eines sozialen Gefüges, oder der Überwachung der physischen und umweltorientierten Bedingungen. Innerhalb dieser Anwendungen können Fälle auftreten, wo Freundschaftsbeziehungen nicht mehr aktuell sind, wo die Kommunikation zweier Endpunkte zusammenbricht, welches mittels einer Unsicherheit des Informationsaustausches zweier Endpunkte ausgedrückt werden kann. Diese Arbeit untersucht die Optimierung des Informationsflusses in Netzwerken, deren Verbindungen unsicher sind, hinsichtlich der Bedingung, dass nur ein Bruchteil der möglichen Kanten für den Informationsaustausch benutzt werden dürfen. Eine eigens entwickelte Datenstruktur - der F-Baum - wird eingeführt, die 2 NP-harte Teilprobleme auf einmal adressiert: zum einen die Berechnung des erwartbaren Informationsflusses und zum anderen die Auswahl der optimalen Kanten. Der F-Baum unterscheidet hierbei unabhängige Zusammenhangskomponenten der wahrscheinlichkeitsbasierten Eingabedaten, deren Informationsfluss entweder analytisch korrekt und effizient berechnet werden können, oder lokal über traditionelle Monte-Carlo sampling approximiert werden können.

Der darauffolgende Abschnitt dieser Arbeit befasst sich mit einem Graphproblem aus Sicht der Optimierungsforschung angewandter Mathematik. Es wird eine neue Variante der Tourenplanung vorgestellt, welches neben kundenspezifischer Zeitfenster auch flexible Zustellstandorte beinhaltet. Darüber hinaus obliegt den Zielorten, an denen Kunden bedient werden können, weiteren Kapazitätslimitierungen. Aus praktischer Sicht ist das VRPTW-FL (engl.: "**V**ehicle **R**outing **P**roblem with **T**ime **W**indows and **F**lexible **L**ocations") eine bedeutende Problemstellung für Paketdienstleister, Routenplanung mit eingeschränkten Stellplätzen oder auch für die praktische Planung der Arbeitsaufteilung von behandelnden Therapeuten/-innen und Ärzten/-innen in einem Krankenhaus. In dieser Arbeit wird für die Bewältigung dieser Problemstellung eine Metaheuristik vorgestellt, die einen hybriden Ansatz mit der sogenannten *Adaptive Large Neighborhood Search* (ALNS) impliziert. Darüber hinaus wird als Konstruktionsheuristik ein *'Backtracking'*-Mechanismus (dt.: Rückverfolgung) angewandt, um initiale Startlösungen aus dem Lösungssuchraum auszuschließen, die weniger vielversprechend sind. In der Evaluierung dieses neuen Ansatz werden Krankenhausdaten untersucht, um auch die Nützlichkeit von flexiblen Zielorten unter verschiedenen Kostenfunktionen herauszuarbeiten.

Im letzten Kapitel dieser Dissertation werden Trends in sozialen Daten analysiert, die Auskunft über die Stimmung der Benutzer liefern, sowie Einblicke in tagesaktuelle Geschehnisse gewähren. Ein Kennzeichen solcher Trends liegt in dem Aufbraußen von inhaltsspezifischen Themen innerhalb eines Zeitfensters, die von der durchschnittlichen Erscheinungshäufigkeit desselben Themas signifikant abweichen. Die Untersuchung der Verbreitung solches Trends über die zeitliche und örtliche Dimension erlaubt es, Trends in Archetypen zu klassifizieren, um somit die Ausbreitung zukünftiger Trends hervorzusagen.

Mit der immerwährenden Skalierung von Graphdaten und deren Komplexität, und den Fortschritten innerhalb der künstlichen Intelligenz, wird das maschinelle Lernen unweigerlich weiterhin eine wesentliche Rolle spielen, um Graphdaten zu modellieren, analysieren und schlussendlich die Wissensextraktion aus derartigen Daten maßgeblich zu fördern.

1

Introduction

Logic is the foundation of the certainty of all the knowledge we acquire.

Leonhard Euler
1707-1783

In today's environment, the systems we are working with have access to many data sources, where the data types are multifarious. It spans the whole space from highly structured and semi-structured data to raw data in text files, video files, or sensory data captured in various applications. With growing computational power and increasing memory capacities, we simultaneously observed the processing of ever-growing complex data and data volume.

In the past, enterprises processed their data from text documentation, image data, or audio files. In recent years, they also had to deal with another wave of unstructured data from social media, streaming data, or data coming from 'smart' devices, which are embedded in the 'Internet of Things' (IoT). According to the current forecasts, the estimation states that 80 percent of worldwide data will be unstructured by 2025. Ironically, even though a company's best asset is its data, it is getting harder to manage it properly. Because today's big data applications often work with unstructured or semi-structured data, we require new tools and methods to analyze them.

Talking about the ever-growing data volume, the *International Data Corporation* (IDC) updated its *Global DataSphere*. It stated that from 2020 to 2025, new data creation will grow at a compound annual growth rate (CAGR) of 23%, resulting in approximately 175 ZB of data creation by 2025 [1]. Moreover, according to one of their projections, *'the amount of data created over the next three years will be more than the data created over the past* 30 *years, and the world will create more than three times the data over the next five years than it did in the previous five'*.

Graph Theory serves as a lifebelt in various real-world settings in this constantly rising data flood. The principle of connecting data in a meaningful way supports decision-making for plenty of business concepts and gains insights into the context within an organization's existing knowledge. Hence, it has become fashionable to examine applications using graph theory in physics,

[1] https://www.seagate.com/files/www-content/our-story/trends/files/idc-seagate-dataage-whitepaper.pdf

chemistry, communication science, electrical and civil engineering, architecture, operations research, genetics, psychology, sociology, economics, anthropology, linguistics, and so forth. Following this trend, it is not surprising that big tech companies like Meta (Facebook), Twitter, Amazon, Microsoft, and Alphabet (Google) invested millions of dollars in creating their own graph data.

From a theoretical point of view, graph theory embraces elements from matrix theory, probability, topology, combinatorics, and group theory, to name a few. Nevertheless, in essence, it encompasses any system involving a binary relation. Generally, graph data can be illustrated in a relatively intuitive schematic representation. Hence, the hurdle in understanding a graph's fundamental nature is rather straightforward, but it takes a lifetime to master the world of complex networks.

This thesis presents novel heuristics and models applied in diverse settings embedded in graph-backed applications. It discusses a novel spectral clustering approach in the scope of homogeneous networks, i.e., networks where entities are considered to be of the same type. A deep learning enhanced representation learning heuristic is proposed for attributed graphs where additional labels are given for the nodes, edges, and graphs, respectively. It also investigates the problem of optimizing the information flow toward a designated source node in probabilistic networks, where connections between nodes are potentially unreliable. Moreover, a metaheuristic solves an extension of a classical optimization problem. Specifically, a vehicle routing problem for which vehicles serve customers within specific time windows and the customers' capacitated delivery locations are flexible. The thesis also analyzes archetypes of trends in social media data to predict the dissemination of information on a global level.

1.1 History of Graph Theory

The earliest recorded occurrence of a graph-related formalization can be found in the work of Leonhard Euler and marks the birth point of Graph Theory [56]. However, the fundamentals were discovered independently in various settings as they naturally arise from applied mathematics. Whereas Euler's observation is based on the physical world, Gustav Robert Kirchhoff investigated graph-related problems in electric networks. Arthur Cayley considered specific structures in organic chemical isomers expressed as graphs. In the following, we will have a brief recap of historical problems that are noteworthy in the context of any graph-related essay.

Königsberg problem.
In 1735, Leonhard Euler (1707-1783) stated the eminent *Königsberg Bridge Problem* [56]. Figure 1.1 shows the connections in Königsberg in Euler's time, whereas Figure 1.2 shows Euler's sketch of it. The vertices can be understood as land sides, whereas edges denote bridges between these pieces of land across the Pregel River. Figure 1.3 shows an abstraction of the problem.

Euler imposed the question to the puzzle as follows: *Could a pedestrian walk around Königsberg cross each bridge only once?* In the language of graph theory, a walk can be interpreted as an alternating sequence of vertices and edges, where the start and end points are vertices. Euler started to formalize his observations and concluded that the Köngisberg problem defines a graph problem being impossible to solve by an *Eulerian path*, where each bridge is visited exactly once.

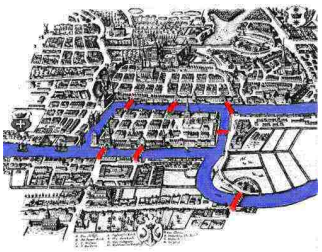


Figure 1.1: *Königsberg in Euler's day*[2]

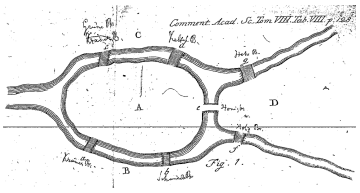


Figure 1.2: *Euler's illustration [56]*

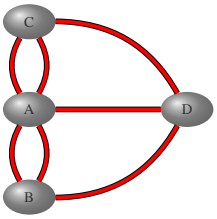


Figure 1.3: *Königsberg's Problem as graph*

Electric Networks.
G. R. Kirchoff (1824-1887) formalized in 1847 the theory of trees for applications in electrical networks to solve systems of simultaneous linear equations [112]. In effect, Kirchhoff replaced each electrical network with its underlying graph and showed that it is not necessary to consider every cycle in the graph of an electric network separately to solve the system of equations. Instead, he showed that independent cycles of a graph determined by any of its 'spanning trees' suffice.

Chemical Isomers.
Ten years later, A. Cayley (1821-1895) formalized trees while trying to enumerate the isomers of saturated hydrocarbons [28]. In this natural setting of organic chemistry, the problem was to find the number of trees with p points in which every point has a degree of 1 or 4.

Four-color conjectures & 'Around the World'.
Two other milestones in graph theory define two essential markers. One was the *four-color conjecture* stating that four colors are sufficient for coloring any map on a plane with the side constraint that countries with common boundaries are colored differently.

The other milestone dates back to a game invented by Sir William Rowan Hamilton (1805-1865) in 1859. The 20 vertices of a regular solid dodecahedron (a polyhedron with 12 faces and 20 corners; each face is a regular pentagon and three edges which meet at each corner) are labeled with names of famous cities. The player's goal is to traverse "*Around the World*" by finding a closed circuit along the edges where each vertex is passed exactly once.

[2]MacTutor History of Mathematics Archive: `https://mathshistory.st-andrews.ac.uk/Extras/Konigsberg/`

1.2 Modern Graph Theory

Over the last twenty-plus years, the growing availability of graph data has given rise to the theoretical formalization for novel characterization of complex real-world graph settings for many new research directions. Some of the formalizations have been well-established in numerous practical applications. However, there is open space for the proliferation of a diversity of network models, including attributive information, heterogeneous information networks, dynamic networks, multi-layer networks, probabilistic networks, and last but not least, *Knowledge Graphs*, among many other task-driven and data-driven models. The recent trends of numerous academic research were majorly driven by the era of big data and the rise of artificial intelligence algorithms, especially insights into modern deep learning architectures.

This thesis provides in § 2 the basic understanding of graph theory. It also highlights graphs' definitions and the implications of how they are used in practical applications. The main chapters introduce novel models and data structures for solving diverse tasks in modern complex networks. § 8.2 also discusses perspectives on how Graph AI will be used in advanced applications in the future and highlights research questions and directions in a broad range of various graph settings.

1.3 Scientific Scope

This dissertation examines novel problems arising in complex networks. It covers research questions spanning the dimension of various graph characteristics, from *Homogeneous Graphs*, *Attributed Graphs*, *Probabilistic Graphs*, to *Spatial Graphs* and *Social Media Graphs*. The toolboxes used to solve those problems range from classical query answering, where a novel data structure is introduced, over modern deep learning approaches in the scope of graph neural networks to tools from operations research (OR).

On homogeneous graphs, this thesis shows an improved, accelerated spectral clustering as a traditional data mining technique where graph structures are used to identify clusters in noisy datasets.

Furthermore, it studies representational learning with graph neural networks where a graph shell attention mechanism captures simultaneously short- and long-term dependencies in attributed graphs. In doing so, the empirical evaluation shows that this novel approach economizes on the number of parameters that have to be trained compared to other state-of-the-art models.

Introducing probabilities on the interactions between entities in a graph spans a new dimension of complexity. This thesis studies the problem of optimizing the information flow in a probabilistic graph given a constraint budget of edges that can be traversed for the information flow. It tackles two NP-hard problems where on the one hand, there is the calculation of the information flow itself and, on the other hand, the proper selection of activated edges. This prob-

lem is of great importance whenever the reliability of a system is of predominant priority, and an information flow towards a designated source node has to be guaranteed.

Next, the thesis examines a vehicle routing problem (VRP) from an operations research (OR) perspective. It originates from a combinatorial optimization problem that aims to identify an optimal set of routes for a fleet of vehicles to serve a given set of customers. It can be seen as a generalization of the traveling salesman problem (TSP), searching for the shortest route of geo-annotated locations. Two dimensions extend the problem presented in this thesis. First, the customers can only be handled in a pre-defined time window (VRPTW). Second, each customer has a set of flexible delivery locations having capacity constraints (VRPTW-FL). That is, there is a preferred location where the costs are reduced w.r.t. a customer, but other possible locations are individually defined, resulting in higher costs. The presented metaheuristic solves the problem by using an *Adaptive Large Neighborhood Search* (ALNS) extended by a *Guided Local Search* (GLS).

Graph Theory entails processing data based on relationships that connect it. The last chapter of this thesis is related to social network analysis, where existing data from a microblogging service is used to answer the specific question, *"Can we identify archetypes to classify trend dissemination"* to get a notion of how data (information) might evolve over time on a global scale. The presented model uses a 4-dimensional tensor as input (flow-source, flow-destination, time, trend). This 4-dimensional tensor can be as well understood as a graph, where the source and destination points are entities in a graph. A relationship labeled by the trend connects the two endpoints. Moreover, the relationship is annotated by temporal information, i.e., a timestamp of the interaction between the two endpoints is given. The spatio-temporal dissemination flow model classifies the archetypes upon the latent features given by a tensor factorization. This yields information to predict the future dissemination of a specific trend.

1.4 Attribution

The author has conceptualized, formalized, and evaluated the scientific works presented in this dissertation. The author has revised the published content, partly restructured and extended it for this thesis. The following paragraphs highlight the author's contributions to the presented publications.

Chapter 3: Homogeneous Graphs
The problems presented in § 3 have been researched in [88, 65]. The work [88] is a joint work with colleagues from the Technical University of Munich and the Aarhus University. The author of this work was involved in conceptualizing and evaluating the findings as well as writing the paper in large parts which originated from this cooperation. The work has been presented by the author on the *'International Conference on Very Large Databases'* [3].

Chapter 4: Attributed Graphs
The collaboration for [65] originated from discussions with a colleague from the Ludwig-Maximilian University of Munich, as well as in cooperation with adjunct professor Dr. Matthias Schubert. The author of this dissertation implemented and evaluated the novel heuristic, conceptualized and formalized the problem with a proof scheme tackling the oversmoothing problem, and finally wrote the paper of this cooperation. The work has been presented by the author on the *'European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases'* [4].

Chapter 5: Probabilistic Graphs
The problem presented in § 5 has been researched in [61, 60] where a patent emerged [63]. The problem was first discussed with colleagues from the former Database System Group at the LMU Munich. Prof. Dr. Matthias Renz contributed in the process of formalizing the problem. The author of this dissertation invented the new data structure, implemented it and run thorough experiments, and was majorly included in the writing process of the papers. An employee from Siemens helped in the patent application. Next to the publication in the top-ranked journal *'Transactions of Knowledge Discovery and Data Engineering'* [5], the work has been presented by the author of this thesis at the *'International Conference on Data Engineering'* [6].

[3] `https://www.resurchify.com/conference_ranking_details.php?id=279`; Ranking: A*
[4] `https://www.resurchify.com/conference_ranking_details.php?id=52`; Ranking: A
[5] `http://portal.core.edu.au/jnl-ranks/415/`; Ranking: A*, Q1
[6] `https://resurchify.com/conference_ranking_details.php?id=271`; Ranking: A*

1.4 Attribution

Chapter 6: Spatial Graphs
A routing scheduling problem is in focus of § 6 and is presented in [66]. The work is a joint work with external researchers and Prof. Dr. Rainer Kolisch from the Technical University of Munich. The work presents an enriched framework implementing an *Adaptive Large Neighborhood Search* (ALNS) being extended by a *Guided Local Search* (GLS). Moreover, in order to find promising initial solutions, a novel backtracking mechanism is presented as construction heuristic. The formalization of the problem builds upon a pilot study presented in [102]. The whole framework had been re-implemented completely by the author of this thesis and a novel evaluation of the problem is presented. Moreover, the formalization of the problem has been partly re-formulated and extended by the author of the thesis compared to the former report. For the ALNS procedure, several novel repair and destroy procedures are proposed and were formalized in the paper. The work is published in the *'European Journal of Operational Research'* [7].

Chapter 7: Social Media Graphs
The problem being discussed in § 7 has been presented in [183]. The work is a joint cooperation with colleagues from the former Database System Group at the LMU Munich and colleagues from the Honk Kong University (HKU). The author of this thesis proposed the usage of a tensor factorization for identifying latent characteristics for the trend dissemination. He also helped in the implementation process of the factorization step and in writing up the theoretical heart of the paper. The work has been presented by a co-author at the *'International Conference on Data Mining Workshops'* [8].

[7] `https://vhbonline.org/vhb4you/vhb-jourqual/vhb-jourqual-3/gesamtliste`; Ranking: A, Q1
[8] `https://www.resurchify.com/conference_ranking_details.php?id=262`; Ranking: A*

1.5 Overview

The top-level division of this dissertation is based on the characteristics of various graph definitions. The first chapter gives a general motivation for the importance of graph theory in modern applications. It also provides a brief history of *Graph Theory* and shows its development from the birth hour up to now. It also encompasses the scientific scope of this thesis and the author's attribution to the presented publications.

§ 2 provides an overview of *General Concepts in Graph Theory*. It introduces various types of networks, and their proper formal definition as practitioners use them in modern applications. Moreover, it presents a graph configurator for classifying various graph types, which is used in the introduction of each main chapter to classify the different research topics.

The main chapters focusing on the scientific contributions start with a description of the context, the purpose, and the scientific contribution of the subsections. The five main chapters are structured as follows:

§ 3 focuses on *Homogeneous Graphs*, i.e., graphs where nodes are not differentiated according to their labels, nor it is distinguished between the various relationships connecting the nodes in a network. In this chapter, a novel spectral clustering approach is presented where the graph structure underneath reflects the inherent structure of the communities.

In § 4 the focus is on representational learning by applying a new graph shell attention heuristic in the message-passing paradigm of *Graph Neural Networks* (GNNs) on *Attributed Graphs*.

§ 5 focuses on *Probabilistic Graphs*. The chapter answers the question how to optimize the information flow to a designated source node within a network, where probability values are attached to the connections between nodes. A novel data structure, called *F-Tree*, is introduced to distinguish between components within a graph for which we can compute the information flow analytically and efficiently and components for which a Monte-Carlo sampling is applied to approximate the information flow.

In § 6, an optimization problem on *Spatial Graphs* defines the theoretical heart. The presented research work introduces a metaheuristic based on an *Adaptive Large Neighborhood Search* (ALNS) where the scheduling of (heterogeneous) vehicles of a fleet serves a set of customers having temporal and spatial constraints. The evaluation shows a practical use-case scenario with therapists in a hospital-related context.

The last § 7 analyses *Social Media Graphs* and provides a model that allows the extraction and characterization of archetypes of trend dissemination flows.

The dissertation concludes in § 8 with a summary of the main results and provides an outlook for future research topics and the practical relevance of Graph AI.

2

General Concepts in Graph Theory

Philosophy is written in that great book which ever lies before our eyes — I mean the universe — but we cannot understand it if we do not first learn the language and grasp the symbols, in which it is written.

Galileo Galilei
1564-1642

In the last decade, Graph data has become ubiquitous in many scientific areas and practical applications: communication infrastructures, drawing and coloring maps, scheduling tasks, and social structures, to name a few. In the previous chapter, the notion of networks has informally been introduced by some historical examples. To study networks in more depth, there is the need to lay a common ground of terminologies that allow to be precise when we speak of distances, neighborhoods, paths in a graph, and so forth. Therefore, at the beginning of every graph journey, one should start with some basic principles and gain a basic understanding of the expected return when defining a network. In parallel to the plethora of applications where graphs are used nowadays, we can also identify various network definitions reflecting the semantics necessary for an application. A fundamental tenet is that graphs provide a *contextualized* understanding of data.

This chapter helps in understanding the basic formulation of networks (see § 2.1) and how they can be represented (see § 2.2). Moreover, in § 2.3, a *Graph Configurator* is presented in order to classify various network types. In subsequent chapters, the graph configurator is used to classify the graphs being covered by the scientific works presented in this thesis.

The definitions and representations in this chapter are not a thorough compilation covering all definitions and characteristics of networks. Instead, it provides the essential understanding to facilitate the step into the main chapters. More task-related definitions are given in the respective contexts of the proposed models.

2.1 Definition of a Graph

This chapter introduces the required terminologies for the essential understanding of graphs.

We will start with an abstract view of the problems' nature this thesis is about. Given any input data, we can represent each data point by any symbol. Moreover, we can represent an interconnection between those symbols by connecting them.

From a *Network Science* point of view, one would use the terminology of *nodes* for the symbols representing the data points, and the connections between them are termed *links*. In *Graph Theory*, the terms *vertices* and *edges* are used in the first place. When we speak of real-world systems, we would rather use *network* terminology, e.g., social networks, transportation networks, the World Wide Web (WWW) is a network of web documents, metabolic networks reflecting chemical reactions, etc. Formally speaking, for example, of the mathematical representation, dealing with the data underneath, or referring to representations used as input for models, we speak in terms of *graph* terminology. However, exceptions can be found, like the definition of the *interest graph*. The terms *networks / nodes / links* and *graphs / vertices / edges* are used interchangeably in the scientific literature. From an Operations Research point of view, the edges are often referred to as *arcs*. In modern applications, the symbols in a network are referred to as *entities* with *relationships* holding between them. Each subsequent chapter provides additional definitions to be clear about the context.

This section provides the formal definitions. Hence, the definitions use the terminology of *Graph Theory*. As the methods and models presented in this thesis are evaluated on real-world scenarios, the thesis carries the practical insinuated title *'Learning from Complex Networks'*.

A graph can be formally defined as follows:

Definition 1. *A **(homogeneous) graph** $\mathcal{G}$ consists of a set $\mathcal{V}$ of **vertices** and a set of **edges** $\mathcal{E}$, for which we write $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Each edge $e \in \mathcal{E}$ is said to join two vertices, which are called its **end points**. If e joins $u, v \in \mathcal{V}$, we write $e = \langle u, v \rangle$. Vertices u and v in this case are said to be **adjacent**. Edge e is said to be **incident** with vertices u and v, respectively.*

The **order** of a graph is equal to the number of its vertices $|\mathcal{V}|$; whereas the **size** of a graph is defined by its number of edges $|\mathcal{E}|$. When we impose certain properties on the set of edges $\mathcal{E}$ the resulting graphs are diverse in their characteristics:

- If $\mathcal{E}$ is *symmetric*, then $\mathcal{G}$ is an **undirected graph**.
- If $\mathcal{E}$ is *symmetric* and *anti-reflexive*, i.e., there are no self-loops, and contains no duplicate edges, then $\mathcal{G}$ is called a **simple (undirected) graph**.
- If $\mathcal{E}$ is *non-symmetric*, then $\mathcal{G}$ is a **directed graph** (or **digraph**).

2.1 Definition of a Graph

A graph can be traversed by following the directions given by the edges. The edges in an undirected graph can be traversed in both directions. When storing the information on the way from one starting vertex to a target vertex, we can come up with the notion of a *walk*:

Definition 2. *A **walk** in a graph is a series of edges (not necessarily distinct):*

$$\langle u_1, v_1\rangle, \langle u_2, v_2\rangle, \ldots, \langle u_p, v_p\rangle,$$

for which $v_i = u_{i+1}$ $\forall i \in [1, \ldots, p-1]$. If $v_p = u_1$ then the walk is closed.

The characteristics of a walk give rise to the terminologies of a *trail, path, cycle*, and *triangle*. Following the notation of Definition 2, we define these terms as follows:

Definition 3. *A **trail** is a walk in which all edges are distinct. A **path** (defined via edges) is a trail in which all u_i are distinct. A* closed path *is called a **cycle** or **circuit**. A graph with no cycles is called **acyclic**. A cylce of length 3 is called a **triangle**. The **length** of a* walk/ trail/ path *is given by p.*

A particular structure that is frequently used in graph analysis is defined as a *tree*:

Definition 4. *A **tree** is a connected graph with no cycles. A **forest** is a union of trees.*

In § 5, a novel data structure is presented for an efficient computation of the information flow within a network. As we will see, the structure of *trees* allows an exact and efficient computation of it. According to *Cayley's formula*, in a graph with n vertices, there are n^{n-2} distinct labeled trees with n nodes (up to isomorphism). In the case of unlabeled trees, there is no known formula but their abundance appears to grow exponentially in n.

Any given vertex in a graph can be adjacent to a set of nodes, called its **neighbors**, connected via incident edges. We define a *neighbor set* as follows:

Definition 5. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. The **neighbor set** $N(u)$ of a vertex $u \in \mathcal{V}$ contains the nodes being adjacent to u, i.e., $N(u) = \{v | v \in \mathcal{V}, \exists e = \langle u, v\rangle \in \mathcal{E}\}$*

A **neighborhood graph** of a central vertex u in a graph $\mathcal{G}$ is also referred to as the *ego graph* of u. The *hop-distance* is measured by the number of edges being traversed from a source node to a target node, i.e., the shortest path length to reach the target. The **degree** of a vertex is the number of edges adjacent to it. In deep learning models building on top of *Graph Neural Networks*, processing the k-hop neighborhood of a vertex is a fundamental property, as it is also discussed in § 4.

A common concept in the field of graph theory is to extract a subset of $\mathcal{V}$ and $\mathcal{E}$. By doing so, we just take a look at a fragment $\mathcal{H}$ of the given graph $\mathcal{G}$, called a *subgraph* of $\mathcal{G}$.

Definition 6. *A graph $\mathcal{H} = (\mathcal{V}', \mathcal{E}')$ is called a **subgraph** of $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ iff $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. The subset $\mathcal{E}'$ is restricted such that $\forall e = \langle u, v\rangle \in \mathcal{E}' : u, v \in \mathcal{V}'$.*

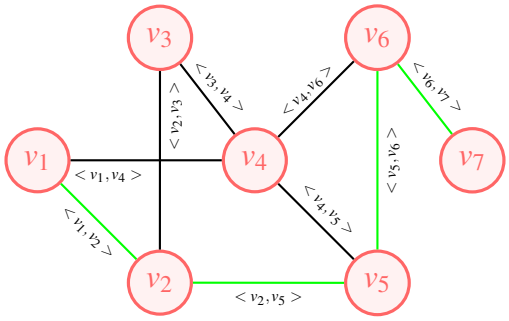


Figure 2.1: *Visualization of a graph with $|\mathscr{V}| = 7$ vertices and $|\mathscr{E}| = 9$ edges. A path from v_1 to v_7 of length 4 is highlighted by green edges*

Subgraphs play an essential role in § 4, where a representational learning model is proposed that relies on processing various sub-parts of a graph. § 5 discusses the decomposition of an input graph in distinctive subgraphs for an efficient transmission of information, and § 6 presents a vehicle routing problem where the vehicles' routes can be understood as subgraphs as well.

2.2 Graph Representation

This thesis uses various representations for networks. Whereas the visual representation is rather intuitive, a prerequisite for understanding the subsequent chapters is to have a basic understanding of fundamental matrices describing graphs.

2.2.1 Visual Graph Representation

A graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ can be visually illustrated as a set of symbols (rectangles, circles, icons, ...) representing the nodes $\mathscr{V}$ and by a set of lines connecting the elements. These lines represent the set of links $\mathscr{E}$. A visual representation of a simple undirected graph with 7 nodes and 9 links is shown in Figure 2.1.

2.2.2 Matrix Representations

An elementary description of a graph is given by a binary square matrix of order $|\mathscr{V}| \times |\mathscr{V}|$:

Definition 7. *Suppose $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ is a simple graph where $|\mathscr{V}| = n$. For $1 \leq i, j \leq n$, we define the square matrix $A = (a_{ij})_{i,j=1}^{n} \in \{0,1\}^{|\mathscr{V}| \times |\mathscr{V}|}$ as the **adjacency matrix** of G which contains the*

following entries:

$$a_{ij} = \begin{cases} 1, & \langle i,j\rangle \in \mathscr{E} \\ 0, & \langle i,j\rangle \notin \mathscr{E} \end{cases}$$

Other important matrix representations of graphs are the *Laplacian*, the *degree*, and the *distance matrices*. The *degree matrix* is a diagonal matrix that contains information about the number of edges attached to each node:

Definition 8. *Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be a simple graph and its adjacency matrix $A \in \{0,1\}^{|\mathscr{V}|\times|\mathscr{V}|}$. For $1 \leq i, j \leq n$:*

$$d_{ij} = \begin{cases} \sum_p A_{ip}, & \text{if } i = j \\ 0 & \text{else} \end{cases} \tag{2.1}$$

*the diagonal matrix $D = (d_{ij})_{i,j=1}^{n} \in \mathbb{N}^{|\mathscr{V}|\times|\mathscr{V}|}$ is the **degree matrix** of $\mathscr{G}$.*

In *Spectral Graph Theory*, the study of *Laplacian* matrices is of special interest as they are main tools for spectral clustering. The *unnormalized graph Laplacian matrix* is defined as follows:

Definition 9. *Let $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ be a graph, $A \in \{0,1\}^{|\mathscr{V}|\times|\mathscr{V}|}$ be the adjacency matrix and $D \in \mathbb{N}^{|\mathscr{V}|\times|\mathscr{V}|}$ be the diagonal degree matrix, then the **(unnormalized) Laplacian matrix** of $\mathscr{G}$ is defined as:*

$$L = D - A \tag{2.2}$$

The spectral characteristics of the Laplacian yield information about the graphs' connectedness. In § 3 a spectral method is presented by analyzing the structural behavior of a weighted graph in order to uncover the clusters in highly noisy datasets.

The Laplacian matrix of Figure 2.1 is:

$$L = D - A = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} 2 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 3 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 4 & -1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 3 & -1 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}$$

Another representation is called the *distance matrix* which is given by a weighted adjacency matrix, i.e., $A \in \mathbb{R}^{|\mathscr{V}|\times|\mathscr{V}|}$ where the weights are given by a weighting function $w : \mathscr{E} \to \mathbb{R}^{>0}$. It contains information pertaining the distance between all connected pair of nodes and is used, e.g., to solve optimization problems like the computation of the shortest path between two nodes. A weighted adjacency matrix is used for the novel spectral clustering approach being presented in § 3, and the metaheuristic used to optimize an extension of the vehicle routing problem in § 6. It is also used in the analysis of trend flows in § 7.

2.2.3 List Representation

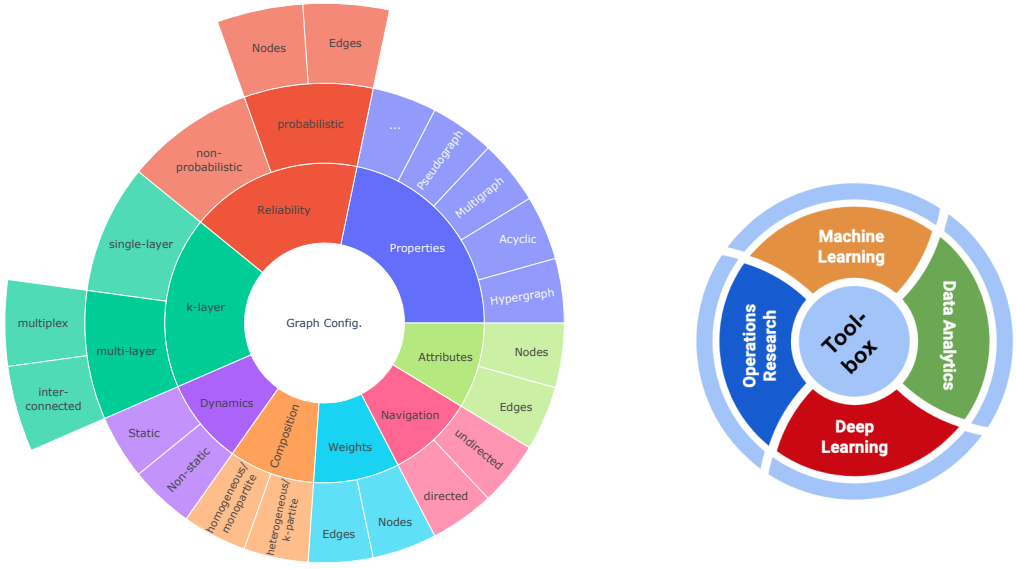
To give the reader a notion about other representation, one can describe a graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ also by its *Adjacency List*. For each node $v_i \in \mathscr{V}$, we keep a listing of its neighbors, i.e., all nodes being adjacent to v_i. The graph's adjacency list of Figure 2.1 is:

$$
\begin{array}{l|l}
v_1 & \rightarrow v_2 \rightarrow v_4 \\
v_2 & \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \\
v_3 & \rightarrow v_2 \rightarrow v_4 \\
v_4 & \rightarrow v_1 \rightarrow v_3 \rightarrow v_5 \rightarrow v_6 \\
v_5 & \rightarrow v_2 \rightarrow v_4 \rightarrow v_6 \\
v_6 & \rightarrow v_4 \rightarrow v_5 \rightarrow v_7 \\
v_7 & \rightarrow v_6
\end{array}
$$

In sparse graph settings, i.e., where $|\mathscr{E}| < |\mathscr{V}|$, the representation via an adjacency list is more memory efficient, as the matrix representations would include mainly '0'-entries.

2.3 Types of Networks in Modern Graph Theory

With the rise of graph-backed applications, the graph community came up with new definitions to include more characteristics being of practical relevance. This thesis encompasses scientific findings from diverse applications using various graph types. In order to categorize graphs, this thesis uses the *Graph Configurator* of Figure 2.2a, spanning an umbrella over the proposed models. The configurator depicts various characteristics in a sunburst visualization. One can define multiple graph characteristics by starting from the inner circles of the visualization, e.g., the dynamics, the composition, the reliability, and so on. The outer segments define a more fine-granular view of the graph at hand and, therefore, specify the graph's information in more detail. For example, regarding the graph's dynamics, we can characterize if the application uses static or temporal graphs; the graph's composition yields information if we have a homogeneous or a heterogeneous setting; or the reliability of interconnections between the graph's entities. One can extend the flexible configurator by more characteristics, and it is used concisely here to depict the scientific contexts of this thesis. Moreover, this thesis uses tools from various domains ranging from classical *Data Analytics*, to *Machine Learning* (ML) techniques, *Deep Learning* (DL), to heuristics originating from the *Operations Research* (OR) (cf. Figure 2.2b). The reader is guided through the main chapters by first specifying the graph's characteristics and the toolbox's domain to get the scientific context at first glance.



(a) Graph Configurator *(b) Toolbox*

Figure 2.2: *Graph configurator and toolboxes being used for the categorization of graphs and classification of methods being used in the thesis*

2.3.1 Graph Types of the Thesis

This chapter provides an overview of various graph settings being in the focus of this thesis and their general definitions. Slightly adaptions to the following definitions are made in respective chapters to fit in the context they are used.

Attributed Graphs.
Naturally, graphs contain useful information simply by their topology and contextual information. In real-world scenarios, we are faced with settings where we have many data sources that may further enrich the information about nodes or connections between them. This leads to settings that are described as *node-attributed graphs* or *edge-attributed graphs*, respectively, depending on which information is provided by the input data. A straightforward way in deep learning is to use the additional attributive information and express it as a vector representation. In doing so, we gain additional information about the nodes and edges in a graph which enhances the performance of a model. An attributed graph can be defined as follows:

Definition 10. *An **attributed graph** is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \sigma, L)$ where $\mathcal{V}$ and $\mathcal{E}$ denote sets of nodes and edges. The domain of attributes is denoted by σ and L is the function that assigns attributes to vertices and edges. Specifically, we use $l(v)$ and $l(e)$ to represent the attributes of vertex $v \in \mathcal{V}$ and edge $e \in \mathcal{E}$, respectively.*

Generally, there are different types of attributed graphs with different types of attributes. Depending on the configuration, one can distinguish between *labeled graphs* (single categorical attributes), *multi-labeled graphs* (multiple categorical attributes), and *property graphs* (multi-dimensional attributes in different domains). This thesis examines in § 3 a novel representation learning model evaluated on real-world examples carrying additional attributive information on the node and edge levels. Hence, we have a *labeled graph*, where each vertex/edge has a single categorical attribute called its *label*. Moreover, the graph itself can have additional information, which we will see for chemical compounds.

Probabilistic Graphs.
Nowadays, the reliability of a system is responsible for its success story. When modeling real-world phenomena, it is likely to incur in situations where the communication between two nodes in a graph is uncertain. The reasons for this unreliability can be manifold, e.g., the communication between sensors might be erroneous or noisy, the connection between two input devices can be disruptive, the flow in a transportation network might be restricted in terms of capacity limitations, the links in a social network might reflect the relative number of communications two persons have, or even intentionally obfuscated for various reasons.

The uncertainty in a graph can be expressed via an additional function attaching to each edge a probability. We define:

Definition 11. *An **probabilistic graph** is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, p)$, where function $p : \mathcal{E} \to (0, 1]$ assigns a probability of existence to each edge.*

In § 5, a novel data structure is introduced for maximizing the information flow towards a designated source node in a probabilistic graph setting. Note that in § 5.2.3, the definition 11 is extended by a function W which maps to each node in the graph an additional information weight.

Spatial Graphs.
In recent years, we observed an increasing availability of GPS-equipped end devices. This also increased the demand for the development of location-based social networking (LBSN) services, e.g., *Google Maps*, *Tripadvisor*, and *Yelp*. Even though location-based information is commonly related to specific research fields examining geographical issues and analyses, the use of spatial information also found its way to graph systems. Hence, several recent approaches now use these geographical features as additional information for analyzing problems in social networks, sensor networks, or logistics (e.g., transportation networks). We can encode spatial information in various ways depending on the use case. This shall give an impression of the flexibility of graph systems which can be adapted individually for certain tasks:

Definition 12. *A **location-location** graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph where nodes $v \in \mathcal{V}$ represent locations and edges $e \in \mathcal{E}$ represent the relation between two locations. Thereby, the semantic*

of the relation can be defined in various ways, e.g., the distance between two locations, the similarity or visits by the same user.

Noteworthy, we see a connection to a *weighted undirected graph*, where the weights can be interpreted as the relationship between two locations. We can also include the information of users into a location-aware graph which can be defined as a bipartite graph, i.e., a graph whose vertices can be divided into two disjoint and independent sets and edges connect only entities between both sets:

Definition 13. *A **user-location graph** $\mathcal{G} = (\mathcal{U}, \mathcal{V}, \mathcal{E})$ is a bipartite graph where nodes $u \in \mathcal{U}$ represent users, node in $v \in \mathcal{V}$ represent locations and edges in $e \in \mathcal{E} \subseteq \mathcal{U} \times \mathcal{V}$ represent relations between users and locations.*

From a semantic point of view, the relation can be flexible, e.g., it may indicate that a user visited or rated a certain location. We can also encode the location information between two nodes on the edges of a graph. The following graph definition gives an idea of this interpretation:

Definition 14. *A **user-user graph** $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph where nodes $u, v \in \mathcal{V}$ represent users and directed edges in $e \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ represent relations between users.*

Some typical edge semantics here may be physical distances, friendship relationships in an LBSN, or features derived from users' location histories (e.g., edges may connect users having visited a common location). The evaluation chapter of analyzing the information flow in probabilistic graphs in § 5 uses data from city datasets. Here, the reliability between two end-points is expressed via the distance of two junctions. Moreover, § 6 presents an optimization heuristic for solving an extension of the famous vehicle routing problem. In this scenario, we are naturally faced with locations where the distance can be expressed as information attached to the links in the graph.

Temporal Graphs.

Real-world phenomena are dynamic by nature, i.e., the interactions between entities evolve over the temporal dimension. The interactions between entities are expressed either by a specific timestamp or a time interval. The temporal information is used in temporal networks where dynamic features are essential. This temporal extension can be found under various terminologies such as evolving graphs, time-varying graphs, timestamped graphs, dynamic graphs, and so on. The interactions between entities can be expressed in two ways, either by attributive information on the links or by an additional function mapping the edges into the temporal domain:

Definition 15. *A **temporal graph** $\mathcal{G} = (\mathcal{V}, C)$ is defined by a set of vertices $\mathcal{V}$ with an associated set of contacts C where each contact $c \in C$ is a triple u, v, t, where $u, v \in \mathcal{V}$ and $t \in T$, where T denotes the temporal domain. Equivalently, a temporal network can be defined as $\mathcal{G}(\mathcal{V}, \mathcal{E}, \tau)$, where τ denotes a function associating each edge to its temporal information $\tau : \mathcal{E} \to T$, where T denotes the temporal domain.*

Notably, this definition can be combined with spatial information ending in spatial-temporal information networks. This thesis handles node-level spatial information and temporal information between entities in § 7, where endpoints contain spatial information and the relationship between them yields temporal information about the interaction between the two entities.

2.3.2 Further Graph Types

The world of graphs is too colorful to be explored on its whole within a single thesis. To give also a notion about other types of graphs and their characteristics which are in the focus of the current research work (time of writing: year 2022), this paragraph presents three other types:

Heterogeneous Information Networks.
In modern applications, we observe that several types of entities interact with each other. Moreover, in real-world scenarios, it can be observed that entities interact in different ways, which results in a pool of semantic interpretations. Whenever we observe that the underlying input data contains more than one node type and relation type, we can define it as a *Heterogeneous Information Network*.

Definition 16. *An **Heterogeneous Information Network** is defined as a directed graph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$ with an object type mapping function $\tau : \mathscr{V} \to A$ and a link type mapping function $\phi : \mathscr{E} \to R$, where each object $v \in \mathscr{V}$ belongs to one particular object type $\tau(v) \in A$, each link $e \in E$ belongs to a particular relation $\phi(e) \in R$, and if two links belongs to the same relation type, the two links share the same starting object type as well as the ending object type. When the types of objects $|A| > 1$ or the types of relations $|R| > 1$, the networks is called a heterogeneous information network; otherwise, it is a homogeneous information network.*

An ongoing research project of the author of this thesis is the relation prediction in Heterogeneous Information Networks and Knowledge Graphs. The problem is framed as an *Reinforcement Learning* problem, where the agent tries to learn topologies of the endpoints to predict the query relationship holding between them including temporal information [64].

Multilayer Networks.
In recent years, one can observe a rise in the interest in modeling interconnected systems to increase connectivity. Basically, it encompasses the interconnectedness of different systems in an infrastructure. This kind of modeling helps, for example, to react more sophisticatedly to random failures compared to single-layer networks (also called *'monoplex'*). Therefore, modeling multilayer networks provide a powerful tool for an in-depth analysis of real-world systems. Generally, the layers display the entities in different interaction contexts where the participation of an entity can occur on various layers. A first observation is that we can distinguish between two types of

edges: the *inter-layer edges* linking instances in different layers and the *intra-layer edges* connecting instances within one layer. If inter-layer edges are defined as the coupling edges which link different instances of the same entity, the model is referred to as a *Multiplex Network*. A *Multilayer Network* is defined as follows:

Definition 17. *Let $\mathscr{L} = \{L_1, \ldots, L_l\}$ be a set of layers and $\mathscr{V}$ be a set of entities. We denote with $\mathscr{V}_l \subseteq \mathscr{V} \times \mathscr{L}$ the set containing the entity-layer combinations in which an entity is present in the corresponding layer. The set $\mathscr{E}_l \subseteq \mathscr{V}_l \times \mathscr{V}_l$ contains the undirected links between such entity-layer pairs. We denote with $\mathscr{G}_L = (\mathscr{V}_L, \mathscr{E}_L, \mathscr{V}, \mathscr{L})$ the multilayer network graph with set of nodes $\mathscr{V}$.*

Multilayer networks have been established as a profound tool to connect metadata to datasets. For example, a 2-layer network can be used to model customer information being curated by administrators whose information is stored on a separate level. In practice, we can observe a trend for graph-backed metadata hubs. Some examples are Airbnb and its Dataportal platform [9], Lyft with Amundsen, and LinkedIn with Datahub. The conflation of various data sources by an additional layer will play a pivotal role in researching multimodality and developing modern applications. Multilayer Networks are essential for architectures including meta-data as discussed in the outlook § 8.2.

Knowledge Graphs.
In 2010 Google introduced its concept of 'Knowledge Graphs' (KGs) to a broader community and has since then been used as a term in computer science articles, papers, and diverse publications. Whereas graphs have always been pervasive as a major research area in Artificial Intelligence (AI), nowadays, there is a need to enable machines to understand and give them the ability to infer new knowledge. Therefore, one of the central intuitions behind Knowledge Graphs was to be able to search for 'things not strings'. With Google being undoubtedly one of the pioneers in this area, other companies started using the concept of representing knowledge as a graph. We can observe even KG-centric startups arising for all kinds of applications such as in government, economic or non-profit, and last but not least, the KG ecosystem emerged as one of the major research fields in cognitive automation systems. An ongoing research project of the author of this thesis is the semantic disclosure of Knowledge Graphs [62].

[9] 'Democratizing Data at Airbnb', talk at GraphConnect, 2017, `https://www.youtube.com/watch?v=gayXC2FDSiA`

3

Homogeneous Graphs

It is not inequality which is the real misfortune, it is dependence.

Voltaire
1694-1778

Attribution

This Chapter uses material from the following publication:

- Ellen Hohma∗, Christian M. M. Frey∗, Anna Beer∗, and Thomas Seidl. Scar: Spectral clustering accelerated and robustified. *Proc. VLDB Endow.*, 15(11): 3031–3044, jul 2022. ISSN 2150-8097. doi: 10.14778/3551793.3551850. URL `https://doi.org/10.14778/3551793.3551850`. [88]

See § 1.4 for an overview of incorporated publications and the author's attribution.
Authors annotated by ∗ contributed equally to the referenced work.

Highlights

- Introducing SCAR, a novel spectral clustering method tackling both, robustness *and* speed;
- Incorporating the Nyström method to accelerate the eigendecomposition in robust spectral clustering;
- Further enhancements for quality and stability of clusterings;
- Throughly, fairly and reproducibly evaluation of the method and comparison to recent state-of-the art methods on established real-world benchmark datasets;

3.1 AI on Homogeneous Graphs

Machine Learning, as a subset of Artificial Intelligence, aims to provide models that learn and improve from data. In the last decade, there have been impressive improvements in applying machine learning to graph-structured data. Generally, the primary objective is to learn suitable representations on the node-/graph-level for certain prediction tasks, discover new patterns, and learn complex dynamics occurring predominantly in graphs with temporal annotations. One common problem in this area is the identification of clusters within a graph which is also referred to as *Community Detection*. From a practical point of view, this is of interest for analyzing social structures or classifying chemical graphs w.r.t. certain graph labels being provided by the input data. Before the rise of deep learning, traditional approaches following standard machine learning paradigms were primarily used to solve the task at hand. This is achieved by incorporating statistics and features defined by heuristic functions or domain knowledge and using a composite of them as input to a standard machine learning classifier. The underlying idea is to apply a shallow clustering technique on the calculated node embeddings. These embedding techniques allow us to project nodes into a vector space where a distance measure representing the similarity between nodes can be defined. After the projection, we can use, for example, distance-based clusterings (e.g., K-Means [141, 140]), density-based clustering methods (e.g., Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [55]), connectivity clustering (e.g., hierarchical clustering [29]) or distribution clustering (e.g., Gaussian mixture [168]). Depending on the clustering heuristic being used, we either get a hard or a soft clustering resulting in *non-overlapping communities* or *overlapping communities*.

This chapter covers learning a graph data representation using a traditional Machine Learning technique. In § 3.2, a graph partition procedure is discussed, based on processing the Laplacian matrix representing the connectivity properties of a graph. Generally, spectral clustering is obtained by applying a standard clustering algorithm (e.g., K-Means) on the eigenvectors of the Laplacian matrix. Therefore, in some sense, spectral clustering can also be seen as a special case of an embedding-based community detection algorithm where the vector representations are given in the spectral embedding.

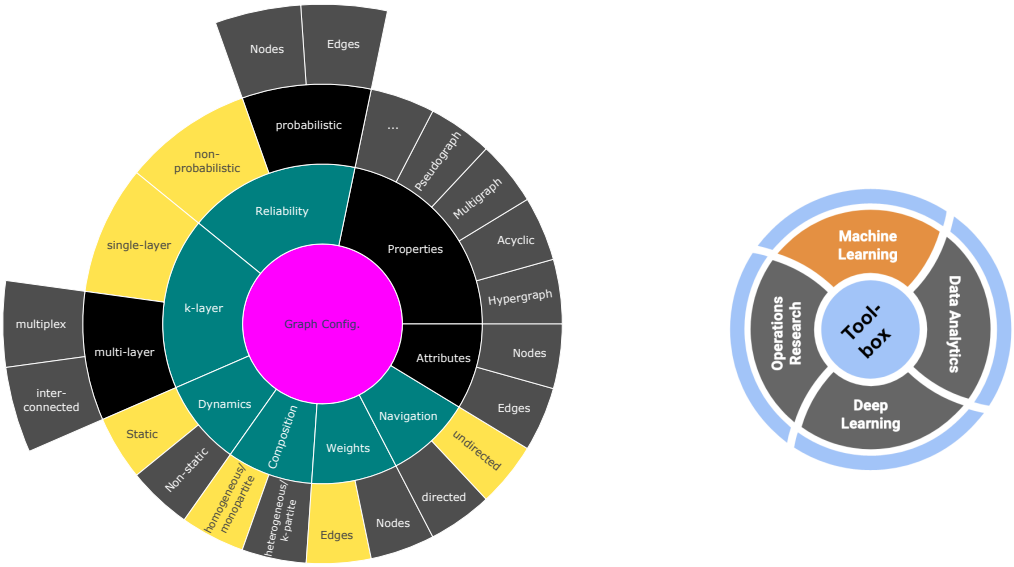
3.2 Robust and Accelerated Spectral Clustering

Spectral clustering is one of the most advantageous clustering approaches. However, standard *Spectral Clustering* [151] is sensitive to noisy input data and has a high runtime complexity. Tackling one of these problems often exacerbates the other. As real-world datasets are often large *and* compromised by noise, we need to improve both robustness and runtime at once. Thus, in this chapter **S**pectral **C**lustering - **A**ccelerated and **R**obust (SCAR), an accelerated, robustified spectral clustering method is proposed. In an iterative approach, it achieves robustness by sep-

arating the data into two latent components: cleansed and noisy data. The eigendecomposition – the most time-consuming step – is accelerated based on the Nyström method. In extensive experiments, SCAR is compared to related recent state-of-the-art algorithms.

SCAR surpasses its competitors in terms of speed and clustering quality on highly noisy data.

Graph Configuration.
The configuration for the following spectral clustering model is shown in Figure 3.1a. The setting is a static, single-layer homogeneous graph with non-probabilistic and undirected edges. While constructing the *k*NN graph, a weighting function is used to express the feature distance between the data samples from the input data. The proposed model is associated to the technical domain of *Machine Learning*.



(a) *Graph Configurator* ***(b)*** *Toolbox*

Figure 3.1: *Graph Configuration and methodology's classification used to solve an accelerated and robustified spectral clustering (SCAR)*

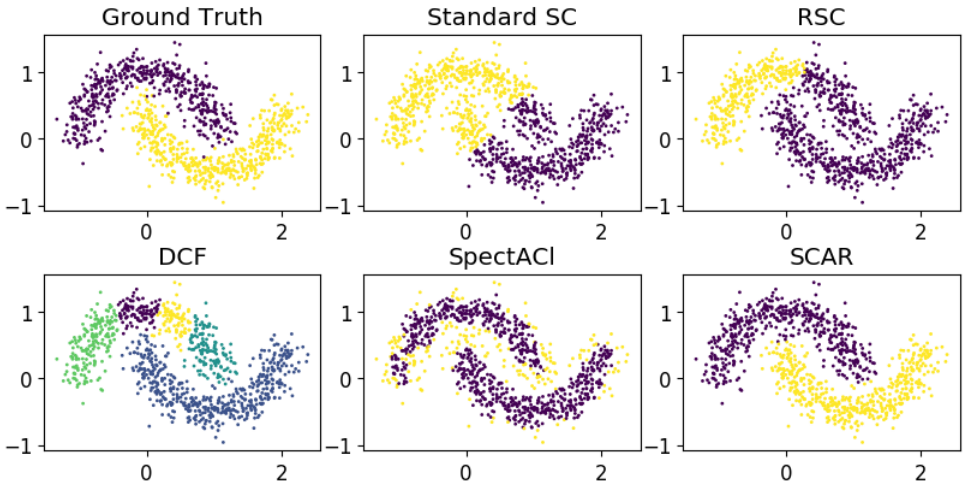


Figure 3.2: *SCAR vs. state-of-the art related clustering algorithms on the moons dataset with noise* = 0.15.

3.2.1 Motivation

Clustering is a fundamental data mining task needed in virtually all areas working with data and also serves as an unsupervised preprocessing step for a plethora of subsequent tasks. One of the most favorable clustering methods is spectral clustering: it is applicable to non-numeric datasets, can find clusters of complex shapes and different densities, and optimizes a mathematically well-defined problem [212]. However, real-world datasets are challenging for several reasons: with newly developed data gathering methods (e.g., in medicine, chemistry, or biology), in recent years datasets grew in dimensionality as well as in size. The runtime complexity of spectral clustering methods is only linear in the number of dimensions, as it works on an affinity graph of the data, making it superior to more traditional clustering methods when working on high-dimensional data. However, the runtime complexity (for the naive implementation) of $\mathcal{O}(n^3)$ w.r.t. the number of data points is comparably large. Furthermore, real-world data often contains noise that is neither handled well by standard spectral clustering methods nor by other clustering methods. Clustering noisy data is in fact a very challenging task, as Figure 3.2 illustrates. It shows a very noisy version of the well-known synthetic moons dataset as clustered by diverse algorithms. It may not come as a surprise that standard Spectral Clustering fails to detect the moons correctly. But also state-of-the-art algorithms that are designed specifically to be robust against noise can only handle noise up to a certain degree and were not able to detect the two clusters correctly. The competitors in Figure 3.2 (as well as in the experiments in Section 3.2.5) are recent clustering algorithms published at high-quality conferences: RSC [20], DCF [201], and SpectACl [84]. The authors of all methods performed extensive experiments showing their superiority against a variety of other clustering methods regarding noise robustness. RSC and DCF also successfully tackle the efficiency problems of clustering. Nevertheless, with the newly

developed method SCAR (**S**pectral **C**lustering - **A**ccelerated and **R**obust), there is a way to even further improve both, clustering quality on highly noisy data *and* efficiency on high-dimensional data.

SCAR uses weighted kNN graphs to capture highly complex structures in the data implying clusters of non-convex shapes. For a good segmentation of the graph, normalized cuts have proven to be desirable [30, 41], suggesting a spectral approach. Based on the concept of RSC [20], SCAR divides the data into a subset containing noise and a subset containing the relevant information for clustering. However, RSC involves the frequent calculation of eigendecompositions in an iterative approach, which is accelerated with the Nyström method. With an elaborated combination of synergistic methods and changes, the proposed model manages to achieve highly competitive results regarding the clustering quality and robustness. In extensive and reproducible experiments the thesis examines and compares its clustering results w.r.t. quality and runtime. SCAR shows the desired behavior for highly noisy datasets, where it outperformed recent state-of-the-art algorithms in quality, noise robustness, and runtime. The model has been evaluated on diverse types of noise and well-known benchmark datasets were used.

Outline. In Section 3.2.2 the thesis gives an overview on related methods. In Section 3.2.3 it explains the basics of the new method. In Section 3.2.4 the thesis introduces the new fast and robust spectral clustering method, called SCAR. In Section 3.2.5 SCAR is evaluated thoroughly, objectively, and reproducibly. Section 3.3 concludes this chapter.

3.2.2 Related Work

Spectral clustering refers to a set of clustering algorithms that partition a given dataset based on the spectrum of the datapoints' affinity matrix. They essentially follow three steps [212]: (1) construct a similarity graph $\mathscr{G}$, (2) compute the Laplacian of $\mathscr{G}$ and its eigendecomposition, and (3) cluster its eigenvectors with a standard clustering method, e.g., k-Means [141, 140]. Spectral clustering surpasses traditional clustering techniques in several aspects: e.g., they find arbitrarily shaped clusters, are applicable on categorical data, solve a clearly defined mathematical goal [212], and can handle varying densities. However, spectral clustering is noise-sensitive [20, 84] and has a relatively high runtime. In the following, the thesis provides an overview of related works in this research field.

3.2.2.1 Improving Runtime

Most recent advances improving any of the steps of spectral clustering can be found in [204]. In the following, the thesis focuses on approaches accelerating the most time-intensive step of spectral clustering, the eigendecomposition. The acceleration is usually achieved with one of two strategies: iteration or sampling.

Iterative approaches. The probably most common method to accelerate the computation of eigenvectors and eigenvalues of a matrix is the *power iteration*. By iteratively multiplying the matrix with a randomly initialized vector (or an estimation of the dominant eigenvector), the eigenvector belonging to the largest eigenvalue is approximated. Generally, the frequent matrix multiplications are expensive, and only the dominant eigenvector can be approximated with the original power method – for spectral clustering, the eigenvectors belonging to the *smallest* eigenvalues are of interest. Note, that the behavior of convergence of iterative approaches usually depends on the distribution and gaps between the eigenvalues [204]. There is a wealth of extensions based on the power iteration aimed at alleviating its downsides for spectral clustering. Using Krylov subspaces allows approximating several eigenvectors at once: E.g., the *Arnoldi iteration* [5] orthogonalizes the vectors spanning the Krylov subspace by applying the Gram-Schmidt process. For Hermitian matrices like symmetric Laplacians, which are used in the process of spectral clustering, the *Lanczos* method has been proposed [122]. The Lanczos method approximates the largest k eigenvectors in $O(|\mathscr{E}|k+|\mathscr{V}|k^2)$ for a graph $\mathscr{G}=(\mathscr{V},\mathscr{E},w)$ [204]. It is used for spectral clustering in [191]. While the Lanczos method performs well even for sparse matrices, it is often prone to numerical instability [26]. The *Implicitly Restarted Lanczos Method* (IRLM) as used, e.g., in ARPACK [129], can reduce numerical instability. Further adaptions involve among others using the inverse matrix to get the smallest eigenvalues and respective eigenvectors [54]. The Krylov-Schur algorithm [195] alleviates additional problems emerging with very large Hermitian or non-Hermitian matrices. As the convergence of symmetric cases depends on the gap ratio of the eigenvalues [156], both ends of the spectrum are approximated, e.g., with *IRLM-BE*.

Sampling-based approaches. Sampling based approaches work on (1) a subset of the edges in the similarity graph or on (2) a subset of the nodes: (1) implies a sparser Laplacian than the original one, while (2) implies a Laplacian of lower dimensionality.

Approach (1) accelerates the eigendecomposition by leveraging matrix operations that are optimized towards sparse input matrices. Working on matrices defined by kNN graphs, choosing a small k leads to sparse matrices that still hold relevant information on the structure of the data. For general graphs, *spectral sparsification* can be applied. It approximates the graph Laplacian with a matrix of same size containing fewer sampled entries. The sampling process ensures that certain pre-defined properties are respected (cf. [204]).

Approach (2) includes *graph coarsening* methods (e.g., [204, 83, 105]) that reduce the original similarity graph to a coarser graph, leading to an adjacency matrix of significantly lower dimensionality. Computing the eigendecomposition on the lower-dimensional matrix and refining it afterwards leads to a significant acceleration.

In the model introduced in Section 3.2.4, it is used the sampling-based *Nyström method*, which is an effective method to significantly speed-up spectral clustering while maintaining good overall eigenvector accuracy (e.g., [59, 223, 134, 214]). The Nyström method has been analyzed, replicated and improved throughout multiple studies: [18], [45], [176], and [231] focus

on the improvements of particular downsides, such as the partial loss of information by sampling landmark points. Furthermore, they provide theoretical evaluations and frameworks on how the quality of the resulting spectral embedding is affected by applying the Nyström approximation. In [164], the impact of the number of landmarks selected as subsample as well as the influence on the overall clustering accuracy is investigated. Thorough studies in [74], [148], and [119] show the impact of sampling techniques picked for identifying the base subset for the Nyström extension. A theoretical analysis of the algorithm's performance and derivations of error bounds are formulated in [31]. The thesis explains the Nyström method in detail in Section 3.2.3.2.

3.2.2.2 Improving Noise Robustness

As spectral clustering has no inherent noise-handling, its quality can suffer from diverse types of noise that often occur in real-world data. In the following, it is distinguished between four different notions of noise that are often mixed up in the literature or not clarified: (1) additional noise points, (2) jitter, (3) noisy features, and (4) noisy edges. Even though they are closely interrelated, they can imply different challenges for (spectral) clustering.

Additional noise points. The probably most common notion of noise is that there are additional noise points in the dataset. They are typically uniformly distributed (and iid) and do not belong to any cluster. E.g., NRSC [138] tackles such noise for spectral clustering by assigning all noise points to an extra cluster. However, they work on the fully connected graph and assume that the majority of edges connected to a noise point has a low weight. AHK [94] also tackles this kind of noise and simultaneously robustify spectral clustering regarding the parameter choice by using an aggregated heat kernel. CAHSM [136] use a hypergraph to compensate for outliers and noise.

Jitter. Adding noise to a dataset can also imply adding a small deviation to each point. E.g., noise adjustment for the moons datasets regulates the deviation from the "perfectly-shaped" moons. A similar effect can be achieved by data quantization. In [95], error bounds for spectral clustering on data with jitter, resp., *perturbed data* are evaluated. Robustness against this type of noise for spectral methods is evaluated, e.g., in SpectACL [84], and RSC [20].

Noisy features. Especially in high-dimensional data, we may encounter noisy features. These refer, for example, to uniformly distributed dimensions of the data that are irrelevant for clustering for at least some points. FWKE-SC [100], SSCG [78], [234], and [10] combine feature weighting with spectral clustering to tackle this problem (similarly to subspace clustering). As they mainly focus on the construction of the similarity matrix, they can be combined with the proposed approach in future work.

Noisy edges. Noise in graphs can also occur as additional edges in the affinity graph of the data. RSC [20] (cf. Sections 3.2.2.3 and 3.2.3.3) focuses on removing edges that connect different clusters, which are also called *corrupted edges*. RSEC [198] regards noisy edges in the context of spectral ensemble clustering. In [11], noise is regarded as "an additive perturbation to

the similarity matrix", including noisy edges as well as corrupted weights of existing edges.

The focus of the proposed model is on robustness w.r.t. noisy edges and jitter. For the other types of noise, filtering additional noise points in a preprocessing step is suggested. For noisy features, the approach can easily be combined with feature weighting approaches that adapt the initial affinity matrix, as SCAR builds on top of the affinity matrix. For weighting the importance of features, one can follow approaches like FWKE-SC [100], using the concept of knowledge entropy, or apply importance scores for attributes that adapt to every point individually, like KISS [16].

3.2.2.3 Comparative Methods

In the experiments in Section 3.2.5 the newly developed method SCAR is compared with standard Spectral Clustering (SC) [151] as well as high-quality state-of-the-art spectral methods that aim at robustness and efficiency: Robust Spectral Clustering (RSC) [20] and SpectACL [84]. Furthermore, the thesis includes the very recently introduced method DCF [201] into its analyses. DCF is not a spectral approach, but also aims at robustness and efficiency.

RSC jointly performs the standard Spectral Clustering and the decomposition of the adjacency matrix A. The latter is assumed to be an additive decomposition of two latent factors, a graph containing corrupted edges and a graph representing the noise-free data. As RSC outperforms basic clustering principles like k-Means and density-based clustering methods on noisy datasets [20], it serves as a baseline in the evaluation in Section 3.2.5.

SpectACL combines approaches from spectral clustering and DBSCAN to solve their major drawbacks regarding noise sensitivity for minimum cut clustering and varying densities for density-based clustering [84]. The core idea is to determine clusters with large average densities while optimizing the density parameters using the spectrum of the weighted adjacency matrix.

DCF aims at improving peak-finding techniques for density-based clustering, which determine groups in a dataset based on their high density as well as distances to clusters of higher density [201]. The approach applies the peak-finding criterion to determine cluster cores instead of point modes, which enables the detection of clusters with varying densities.

3.2.3 Preliminaries

In the following, the thesis gives some preliminary basics for the proposed method SCAR. In Section 3.2.3.1, it is clarified the notation used throughout the description of the novel model. In Section 3.2.3.2 the thesis explains the Nyström method that is used to accelerate the eigendecomposition in detail. In Section 3.2.3.3 it is elaborated on the robustificaton method being incorporated in the method SCAR.

3.2.3.1 Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ be an undirected, weighted graph where $\mathcal{V}$ denotes a set of nodes, $\mathcal{E}$ denotes a set of edges connecting nodes, and w denotes a weight function on the edges $w : \mathcal{E} \to \mathbb{R}^{>0}$. Its adjacency matrix $A \in \mathbb{R}^{n \times n}$ is defined by its entries a_{ij} with $a_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in \mathcal{E}$, else $a_{ij} = 0$. Let $D := \text{diag}(\deg(v_1), \ldots, \deg(v_n)) \in \mathbb{R}^{n \times n}$ be the degree matrix of $\mathcal{G}$ where $\deg(v_i) := |\{v_j \in V \mid (v_i, v_j) \in \mathcal{E}\}|$ is the degree of node v_i. The Laplacian L of $\mathcal{G}$ is defined as $L := D - A$. The Laplacian L is symmetric and positive-semidefinite in $\mathbb{R}^{n \times n}$. Hence, the n eigenvalues $\Lambda = [\lambda_1, \ldots, \lambda_n]$ of L are real and positive. The associated eigenvectors are denoted by $H = [h_1, \ldots, h_n]$, resp., the approximated eigenvectors by $\hat{H}$. Furthermore, $\mathcal{X} = \{x_i\}_{i=1}^n$ denotes the set of n input data samples, where $x_i \in \mathbb{R}^d$ is a d-dimensional feature vector.

3.2.3.2 Nyström Method for Eigenvector Approximation

The Nyström method has shown great promise in existing literature to speed-up the eigenvector calculation (e.g., [59, 223, 134, 214]). To accelerate the eigenvector computation, only a subsample of the whole dataset is used. A matrix $M \in \mathbb{R}^{n \times n}$ can be partitioned into:

$$M = \begin{bmatrix} M_1 & M_2{}^T \\ M_2 & M_3 \end{bmatrix}, \tag{3.1}$$

where $M_1 \in \mathbb{R}^{m \times m}$ represents the affinities between m sampled points in the subset, $M_2 \in \mathbb{R}^{(n-m) \times m}$ contains all weights from the $n - m$ remaining points to the m subsampled points and $M_3 \in \mathbb{R}^{(n-m) \times (n-m)}$ captures the remaining affinities between all points not chosen for the subset. After choosing landmark points for the approximation, the eigenvectors H_1 of M_1 can be calculated. The thesis introduces diverse eigendecomposition approaches that can be used in Section 3.2.2.1 and compares them empirically in Section 3.2.5.6.2.

Using the Nyström extension [59], we can extrapolate the eigenvectors for all remaining points. Let H and Λ be the eigenpairs of M, it follows:

$$M = H\Lambda H^T = \begin{bmatrix} H_1 \\ H_2 \end{bmatrix} \Lambda \begin{bmatrix} H_1 \\ H_2 \end{bmatrix}^T = \begin{bmatrix} H_1 \Lambda H_1^T & H_1 \Lambda H_2^T \\ H_2 \Lambda H_1^T & H_2 \Lambda H_2^T \end{bmatrix} \tag{3.2}$$

Thus, if H_1 denotes the eigenvectors for the subsampled points M_1, we can deduce H_2, the eigenvectors for all remaining points, with $H_2 = M_2 H_1 \Lambda^{-1}$. Sorting the extrapolated eigenvectors for the remaining points back into the calculated eigenvectors for points chosen as subsample yields the approximated eigenvectors $\hat{H} \in \mathbb{R}^{n \times m}$ for M:

$$\hat{H} = \begin{bmatrix} H_1 \\ M_2 H_1 \Lambda^{-1} \end{bmatrix} \tag{3.3}$$

In the last step, the approximated eigenvectors $\hat{H}$ are orthogonalized. By using only a subsample of the data, the time complexity can be reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2 + m^3)$, where usually $m \ll n$ [135].

3.2.3.3 Robustifying Spectral Clustering

In order to robustify spectral clustering, the model follows RSC [20]. The main idea is to separate the input graph with adjacency matrix A into two latent subcomponents described by A^g and A^c:

$$A = A^g + A^c \tag{3.4}$$

A^c reflects the **c**orrupted edges in the graph and A^g contains only the noise-free, "**g**ood" edges. The partitioning into two segments can be determined and improved by independently optimizing the spectral embedding for each subgraph. In practice, it is sufficient to resolve only one component, since its counterpart can easily be deduced from the original representation (see Equation 3.4). In [212], it has been shown that spectral clustering can be transformed into a trace minimization problem for A. Following this idea, in [20], the authors proved that the solution to A^c can be attained by solving a maximization problem for $Tr(H^T L(A^c)H)$, where $L(A^c)$ denotes the Laplacian of matrix A^c. The corresponding objective function for the unnormalized Laplacian (cf. [20]) is defined as:

$$f([a^c_e]_{e\in\mathcal{E}}) := \sum_{(v_i,v_j)\in\mathcal{E}} a^c_{i,j} \cdot \|h_i - h_j\|_2^2 \tag{3.5}$$

Further constraints are given by θ and m. The parameter θ denotes the maximum number of global corruptions that are deleted: $|\{(v_i,v_j)|a^c_{ij} \neq 0\}| \leq 2 \cdot \theta$. The parameter m implies the minimum number of nodes that each node in A^g is connected to: $|\{v_j|a^g_{ij} \neq 0\}| \geq m \cdot deg(v_i)$ for each node v_i.

To solve the maximization problem in order to find edges which should be assigned to A^c, the models use a greedy approach that is described in [128]. The idea is to sort all edges $e \in \mathcal{E}$ in descending order according to their scores p_e being defined as:

$$p_e = p_{ij} = a_{ij} \cdot ||h_i - h_j||_2^2 \tag{3.6}$$

The model iteratively adds edges to A^c such that the side constraints defined by parameters θ and m are fulfilled. Further details, proofs, and the reduction to the multidimensional knapsack problem [159] can be found in [20].

3.2.4 SCAR - Spectral Clustering Accelerated and Robust

The thesis proposes a new clustering method SCAR (**S**pectral **C**lustering – **A**ccelerated and **R**obustified). SCAR separates the affinity graph of the data in an iterative approach into two latent components: a clean graph, which is used for the subsequent clustering, and a graph containing noisy edges. Likewise to Robust Spectral Clustering (RSC) [20], it detects noisy edges in each step that are disadvantageous for clustering. Therefore, it reaches overall robustness against noise compared to the original spectral clustering [151]. SCAR is significantly faster than RSC as it accelerates the most time-intensive step, the eigendecomposition, using the Nyström method [59] explained in Section 3.2.3.2. Furthermore, several aspects of RSC were improved significantly, such that SCAR is not only faster but also achieves remarkably better results in real-world experiments as shown in Section 3.2.5. Figure 3.3 gives an overview of SCAR and highlights the most important steps that deviate from RSC. In the following, it is described each step of the novel method in more detail. Algorithm 1 shows the corresponding pseudocode.

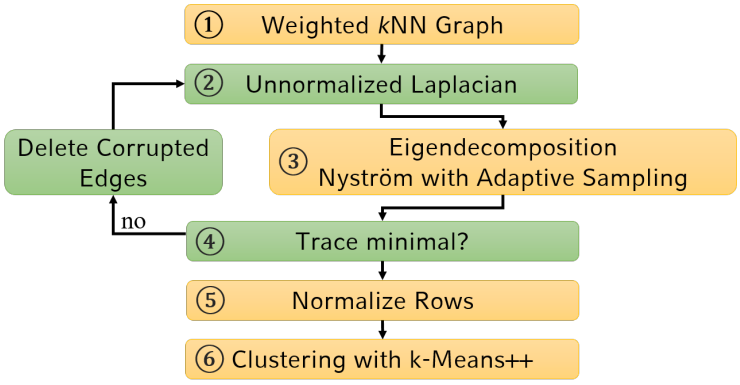


Figure 3.3: *Overview of SCAR. Green boxes imply steps in the method that are analogue to RSC [20] and orange implies a significant change or addition.*

Step 1 The procedure calculates the symmetric, weighted kNN graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, w)$ of the input data $\mathcal{X}$ (cf. line 1 in the pseudo-code). Each data point $x_i \in \mathcal{X}$ implies a node $v_i \in \mathcal{V}$, where $|\mathcal{X}| = |\mathcal{V}| = n$, i.e., there exists a bijective mapping $\phi : \mathcal{X} \to \mathcal{V}$. Further, $\mathcal{E} = \{(v_i, v_j) \mid \forall v_i, v_j \in \mathcal{V}, i \neq j : v_i \in kNN(v_j) \vee v_j \in kNN(v_i)\}$, and the weight of edges is defined by the Gaussian similarity function:

$$w(v_i, v_j) = exp\left(\frac{-||\phi^{-1}(v_i) - \phi^{-1}(v_j)||^2}{2\sigma^2}\right) \tag{3.7}$$

In the experiments, $\sigma = \sqrt{nd/2}$ is used per default. kNN graphs are suitable to find clusters of arbitrary shapes and varying density. Thus, when using spectral clustering, they are on most real world datasets superior to fully connected graphs (FCG), ε-neighborhood graphs, or Gabriel

Graphs [96]. The evaluation in Section 3.2.5 supports these findings. In contrast to RSC [20], SCAR uses a weighted kNN graph and apply the Gaussian kernel to weight the edges, which gives more weight to closer points than to points farther away. This further improves clustering results in general [151], as it offers more information that can be relevant for clustering.

Step 2 As elaborated in [85], the unnormalized Laplacian is more suitable than normalized versions to discern between clusters and outliers, resp., noise in the eigenspace, amplifying the difference between corrupted and clean edges later. Thus, the unnormalized graph Laplacian (line 4) is calculated as $L = D - A$ based on $\mathscr{G}$.

Step 3 The model approximates the eigenvalues of L with the Nyström method as described in Section 3.2.3.2. Following [148], an adaptive sampling approach is used, where $\alpha \cdot n$ points with the highest degrees are chosen as landmarks (cf. line 6). As noise points are unlikely to be nearest neighbors of nodes outside of their own neighborhood (compare to, e.g., ideas of outlier detection algorithms ODIN [82] or kNN-LOF [225]), their corresponding nodes in the kNN graph tend to have lower degrees. As we sample the nodes with the highest degrees, potential losses regarding the set of edges concern prevalently the noisy edges, we want to remove anyway. The procedure then approximate the first k eigenvectors on $L_1 \in \mathbb{R}^{(\alpha n)\times(\alpha n)}$, which is a small submatrix of $L \in \mathbb{R}^{n\times n}$ (see Equation 3.1). The resulting matrix $\tilde{H}_1 \in \mathbb{R}^{\alpha n\times k}$ contains the first k approximated eigenvectors of L_1. The sampling of the submatrix is outlined in lines 7-10, whereas line 11 shows the eigendecomposition. In line 12 the Nyström extension is carried out. In Section 3.2.5 the thesis thoroughly investigates several decomposition methods for computing the eigenpairs of L_1. In the following, the framework works on $\tilde{H} \in \mathbb{R}^{n\times k}$, as obtained by Equation 3.3.

Step 4 SCAR checks in line 15 of the pseudo-code whether the trace of $\tilde{H}^T L(A^g)\tilde{H}$ has decreased compared to the previous iteration. The identification and extraction of corrupted edges in the graph is described in Section 3.2.3.3 and follows the approach of RSC [20]. SCAR applies Equation 3.6 in line 19, i.e., p_{ij} is calculated for all edges $(v_i, v_j) \in \mathscr{E}$. High values for p_{ij} indicate a high dissimilarity between node embeddings of v_i and v_j, even though the nodes are connected. Thus, assigning an edge (v_i, v_j) with a high value p_{ij} to the noise component A^c improves the clustering quality as the edge is disregarded in the subsequent clustering step.

However, to ensure sparsity thresholds, bounds set with θ and m are respected [20]. The parameter θ prevents eliminating too many edges required for reasonable clustering results by limiting the maximum number of overall removable edges. The parameter m ensures a maximum local sparsity, i.e., each node keeps at least a portion m of its originally connected edges. In the experiments, $m = 0.5$ is used per default. If updates on the graph separation still lead to quality improvements, SCAR recalculates L and approximates its eigendecomposition again with the Nyström method. The procedure alternatingly updates A^g in line 22 and $\tilde{H}$ until the trace cannot be significantly lowered.

Algorithm 1: SCAR Algorithm

Input: Dataset X, user input k, nn, α, θ, m
Output: Clustering containing assigned labels

1 $A \leftarrow$ kNN_graph(X, nn);
2 $A^g \leftarrow A$;
3 **for** $iter < max\_iterations$ **do**
4 $\quad L \leftarrow$ Laplacian(A^g) ; `// see Section 3.2.3.1`
5 $\quad$ `/* Nyström method */`
6 $\quad X_l \leftarrow \alpha \cdot |X|$ landmarks;
7 $\quad i \leftarrow$ indices_of(X_l);
8 $\quad j \leftarrow$ indices_of($X \backslash X_l$);
9 $\quad L_1 \leftarrow L[i,i]$;
10 $\quad L_2 \leftarrow L[j,i]$;
11 $\quad \tilde{H}_1, \Lambda \leftarrow$ eigendecomposition(L_1);
12 $\quad \tilde{H}_2 \leftarrow L_2 \tilde{H}_1 \Lambda^{-1}$; `// see Equation 3.3`
13 $\quad \tilde{H} \leftarrow$ reassemble($\tilde{H}_1, \tilde{H}_2$);
14 $\quad trace \leftarrow$ sum(Λ);
15 $\quad$ **if** *trace is minimal* **then**
16 $\quad\quad$ break;
17 $\quad$ **end if**
18 $\quad$ `/* Removing corrupted edges */`
19 $\quad p_{i,j} \leftarrow a_{i,j} \cdot \|h_i - h_j\|_2^2$; `// see Equation 3.6`
20 $\quad removed\_edges \leftarrow$ edges(argmax(p), θ, m);
21 $\quad A^c \leftarrow$ matrix($removed\_edges$);
22 $\quad A^g \leftarrow A - A^c$;
23 **end for**
24 $\bar{H} \leftarrow$ row-wise_norm($\tilde{H}$);
25 $Clustering \leftarrow k$_means++(rows($\bar{H}$));

Step 5 As suggested by [20, 31, 59], the procedure orthogonalizes and norms the first k resulting approximated eigenvectors row-wise (cf. line 24) which increases clustering quality and stability:

$$\bar{H}_{[i,:]} = \frac{\tilde{H}_{[i,:]}}{\|\tilde{H}_{[i,:]}\|_2} \tag{3.8}$$

Step 6 In the last step (shown in line 25), the model clusters the first k rows of $\bar{H}$ (that has the eigenvectors of A^g as columns) with k-Means++ [6]. k-Means++ improves the selection of initial cluster centers for k-Means, leading to an earlier convergence and thus further speed-up compared to traditional spectral clustering approaches using k-Means.

3.2.5 Evaluation

In the following, the thesis examines the method SCAR thoroughly. In Section 3.2.5.1, the experimental setup is presented. In Sections 3.2.5.2 and 3.2.5.3 the thesis analyzes SCAR's clustering quality, noise robustness, efficiency and scalability. In Section 3.2.5.4 a summary of SCAR's clustering and speed performance and regard their mutual dependencies is presented. In Section 3.2.5.5 the thesis evaluates the improvements of SCAR over RSC and SC. In Section 3.2.5.6 the thesis evaluates the influence of various hyperparameters and design choices. SCAR retained an excellent balance between speed and quality over all experiments, while we refrained from hiding experiments that did not deliver desirable results in order to prevent overoptimism [21].

3.2.5.1 Experimental Setup

Datasets. In the evaluation, two synthetic datasets and ten real-world benchmark dataset are used. Both synthetic datasets, moons and circles, are constructed using data generator functions from the scikit-learn library.

Real world benchmark datasets *iris*, *dermatology*, *banknote*, *pendigits*, and *Letter Recognition* (*letters* for short) were obtained from the *UCI- MLR*[10]. *MNIST* and *USPS* were obtained from the repository of the *CS NYU*[11]. Similar to the work of [20], random subsamples were selected for the MNIST dataset. For the pendigits dataset, specific subsets pendigits_{16} and pendigits_{146} were defined as benchmark datasets in the literature [20, 94, 138]. For *dermatology* the feature about the *age* of patients is omitted as the dataset is incomplete w.r.t this feature. The data statistics are summarized in Table 3.1.

Competitors. The thesis compares SCAR with standard Spectral Clustering (SC) [151][12], Robust Spectral Clustering (RSC) [20][13], normalized SpectACl [84][14], and Density Core Finding (DCF) [201][15].

Implementation Details. SCAR is implemented in Python, building off of the implementation of RSC [20][6]. Additionally, the libraries scikit-learn, NumPy, Scipy and slepc4py/petsc4py [16] were used. Experiments were run on an Intel(R) Xeon(R) Silver 4208 CPU @ 2.10GHz using 32GB RAM.

Code: available on GitHub [17]

Hyperparameter Setting. For the synthetic datasets the default value of 0.15 is used for the

[10] `https://archive.ics.uci.edu/ml/index.php` (retrieved: Feb 18, 2022)

[11] `https://cs.nyu.edu/~roweis/data.html` (retrieved: Feb 25, 2021)

[12] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html (last accessed: Jul 14, 2022)

[13] `https://github.com/abojchevski/rsc`(last accessed: Jul 14, 2022)

[14] `https://bitbucket.org/Sibylse/spectacl/src/master/` (last accessed: Jul 14, 2022)

[15] `https://github.com/tobinjo96/DCFcluster` (last accessed: Jul 14, 2022)

[16] `https://slepc.upv.es/documentation/` (last accessed: Jul 14, 2022)

[17] `https://github.com/SpectralClusteringAcceleratedRobust/SCAR.git`

Table 3.1: Dataset properties used in the analysis.

	dataset	n	d	k	noise [%][18]	LB-UB [%][19]
syn.	moons	1,000	2	2	15	
	circles	1,000	2	2	15	
real	iris	150	4	3	7	5-9
	dermatology	366	33	6	9	4-14
	banknote	1,372	4	2	2	0-4
	pendigits_{16}	1,499	16	2	1	0-2
	pendigits_{146}	2,279	16	3	1	0-2
	pendigits	7,494	16	10	9	2-13
	USPS	11,000	256	10	24	12-33
	MNIST-10K	10,000	784	10	24	13-29
	MNIST-20K	20,000	784	10	21	11-27
	letters	20,000	17	26	46	20-61

parameter *noise* that regulates the jitter. Note that this value is significantly higher than the values applied in, e.g., RSC [20]. Let us tune $\alpha \in [0.1, 0.2, \ldots, 0.9]$. For every dataset, the parameter θ is fixed in all the experiments dataset-specific, where $\theta \in \{20, 30, 200, 500, 1k, 10k, 30k, 60k\}$. Following the rule-of-thumb popularized by [46], the thesis uses $2\sqrt{n}$ as an upper bound for *nn* and tested values in 10 percent steps for all methods, accordingly. For a fair comparison to the competitor *DCF*, it is also evaluated the parameter β used in their method in the range of $[0.1, 0.2, \ldots, 0.9]$ to obtain best scores for the cluster metrics.

3.2.5.2 Clustering Quality

Clustering quality is measured using the *Normalized Mutual Information* (NMI) [196] and *Adjusted Rand Index (ARI)* scores, which range from 0 to 1. Higher values imply a better accordance to the ground truth. Following the suggestion of [171], ARI should be used when the reference clustering has large equal sized clusters; scores based on mutual information should be used when the reference clustering is unbalanced and there exist small clusters. In the following, all experiments were run for 10 trials and the average clustering scores per parameter setting are reported if not stated otherwise.

3.2.5.2.1 Effectiveness

In Table 3.2 on the left, the thesis summarizes the best NMI and ARI scores evaluated on each dataset. In order to obtain the best outcomes for each dataset and each method, a grid-search over the respective parameter spaces is applied as outlined in Section 3.2.5.1. The dependence of NMIs on the number of neighbors *nn* can be seen in Figure 3.4. The thesis discusses the runtimes

[18] for synth. datasets noise is added in the sklearn function as standard deviation of Gaussian noise, for real datasets, noise is measured as ratio of inter-cluster edges vs. total edges in the *k*-Graph for all tested *nn*

[19] LB = noise lower bound, UB = noise upper bound

shown in the right part of Table 3.2 – also in combination with the quality of the clusterings – in Section 3.2.5.3 and analyzes the influence of hyperparameter settings in Section 3.2.5.6.

SCAR reaches on average the best NMI/ARI scores while those results were reached on average with the second best runtime of all tested algorithms. SCAR's average runtime is approximately an order of magnitude faster compared to the original standard spectral clustering algorithm (SC) and DCF. SCAR always returns clusterings of solid quality, in contrast to, e.g., SpectACl, which is not able to find an acceptable clustering for the datasets banknote or subsets of the pendigits dataset (marked in red in Table 3.2, see also Figure 3.4). Second best values were often reached by SC, which is, however, not designed to reach fast runtimes. SC's good results on the benchmark datasets confirm the high potential of spectral methods for high-quality clustering results. Further, we observe that SCAR can handle highly noisy datasets like moons, where SC as well as RSC could not correctly detect the clusters, reaching NMIs (ARIs) of only 0.43 (0.72). The thesis regards the sensitivity of all methods w.r.t. the parameter *nn* in Figure 3.4. Where most methods are rather robust w.r.t. the parameter *nn*, their default settings may not be optimal: in Figure 3.2, all algorithms' default parameter settings are applied on the moons dataset. Here, none of the competitors could find the clusters correctly. In contrast, optimized parameter settings w.r.t. the NMI/ARI via a grid search is shown for Table 3.2. Furthermore, the banknote dataset was perceived as an interesting case, as SCAR significantly surpassed its competitors. SpectACl, e.g., was not able to produce any meaningful clustering results over a variety of tested parameter settings, reaching a maximum NMI (ARI) of 0.02 (0.03). All other competitors reached NMI/ARI scores around 0.62. The banknote dataset contains 4-dimensional representations of forged and authentic banknotes. Its clusters overlap in all dimensions, making its similarity graph highly noisy. Thus, the advantages of SCAR's noise robustness can be seen here, yielding an outstanding NMI (ARI) of **0.86** (**0.90**) for the proposed method.

Even though SCAR yields very good results for most datasets, there is still room to further improve clustering results on high-dimensional datasets in future work. Especially, performance on datasets emerging from pixel-data (USPS and MNIST versions) could benefit from applying feature weighting approaches as outlined in Section 3.2.2. Table 3.2 shows that despite their different strengths, the clustering metrics do not differ much in how the investigated methods compare. Thus, only NMI is reported in the following as the default metric.

Table 3.2: Maximum clustering quality reached, measured by normalized mutual information (NMI) scores and adjusted rand index (ARI), as well as minimum runtimes (in seconds) reached for best NMI scores and overall. Best/Second-best results are **bold**/underlined. Values regarded closer in the text are marked in red for faster readability.

		max NMI ARI										min runtime of best NMI (min runtime overall)				
	dataset	SC		RSC		DCF		SpectACl		SCAR		SC	RSC	DCF	SpectACl	SCAR
syn.	moons	0.43	0.72	0.43	0.72	0.91	0.96	0.88	**0.98**	**0.92**	0.96	0.15 (0.11)	0.19 (0.14)	0.14 (0.13)	0.11 (0.08)	**0.06** (**0.03**)
	circles	0.00	0.00	0.19	0.08	0.33	0.16	**0.79**	**0.86**	0.50	0.57	0.13 (0.11)	0.32 (0.20)	0.09 (0.07)	0.07 (0.06)	**0.05** (**0.03**)
real	iris	0.82	0.83	0.81	0.75	0.77	0.75	0.76	0.73	**0.84**	**0.85**	**0.03** (**0.02**)	0.04 (0.04)	0.08 (0.06)	0.06 (0.04)	**0.03** (**0.02**)
	dermatology	**0.93**	0.91	**0.93**	**0.92**	0.91	0.88	0.88	0.88	0.89	0.89	**0.03** (**0.03**)	0.09 (0.05)	0.09 (0.08)	0.08 (0.08)	0.05 (0.04)
	banknote	0.61	0.62	0.61	0.62	0.62	0.62	0.02	0.03	**0.86**	**0.90**	0.16 (0.15)	0.35 (0.19)	0.11 (0.09)	**0.10** (0.10)	0.12 (**0.03**)
	pendigits_{16}	**0.92**	**0.95**	0.90	0.94	0.78	0.76	0.22	0.10	0.90	0.94	0.26 (0.18)	0.37 (0.21)	**0.13** (0.12)	0.17 (0.14)	**0.13** (**0.08**)
	pendigits_{146}	0.95	0.96	**0.96**	**0.97**	0.87	0.86	0.70	0.58	0.95	**0.97**	0.41 (0.41)	0.87 (0.69)	0.29 (0.26)	0.29 (0.29)	**0.27** (**0.17**)
	pendigits	0.81	0.67	0.82	0.67	**0.84**	**0.76**	0.74	0.59	0.82	**0.76**	3.88 (2.94)	8.25 (4.05)	**0.96** (**0.80**)	2.09 (1.73)	2.68 (1.38)
	USPS	0.65	0.46	**0.68**	0.45	0.60	0.31	0.58	0.42	0.63	**0.48**	22.22 (22.22)	10.33 (9.70)	55.42 (54.89)	**4.00** (3.86)	4.59 (**3.18**)
	MNIST-10K	0.67	0.50	**0.74**	**0.55**	0.59	0.45	0.62	0.50	0.61	0.44	36.29 (36.29)	10.49 (10.49)	114.03 (111.82)	**5.00** (4.91)	7.34 (**4.41**)
	MNIST-20K	0.68	0.51	**0.76**	**0.55**	0.62	0.49	0.63	0.49	0.60	0.52	244.87 (244.87)	46.45 (31.39)	444.92 (385.94)	**21.18** (**21.18**)	38.83 (**21.18**)
	letters	0.42	0.16	0.42	0.13	**0.56**	0.17	0.38	0.12	0.46	**0.22**	418.02 (62.48)	38.29 (38.29)	**8.94** (**8.91**)	13.88 (12.99)	19.06 (10.84)
	Avg.	0.65	0.60	0.68	0.61	0.70	0.59	0.6	0.52	**0.74**	**0.70**	60.53 (30.81)	9.67 (7.95)	52.1 (46.93)	**3.91** (3.78)	6.10 (**3.44**)

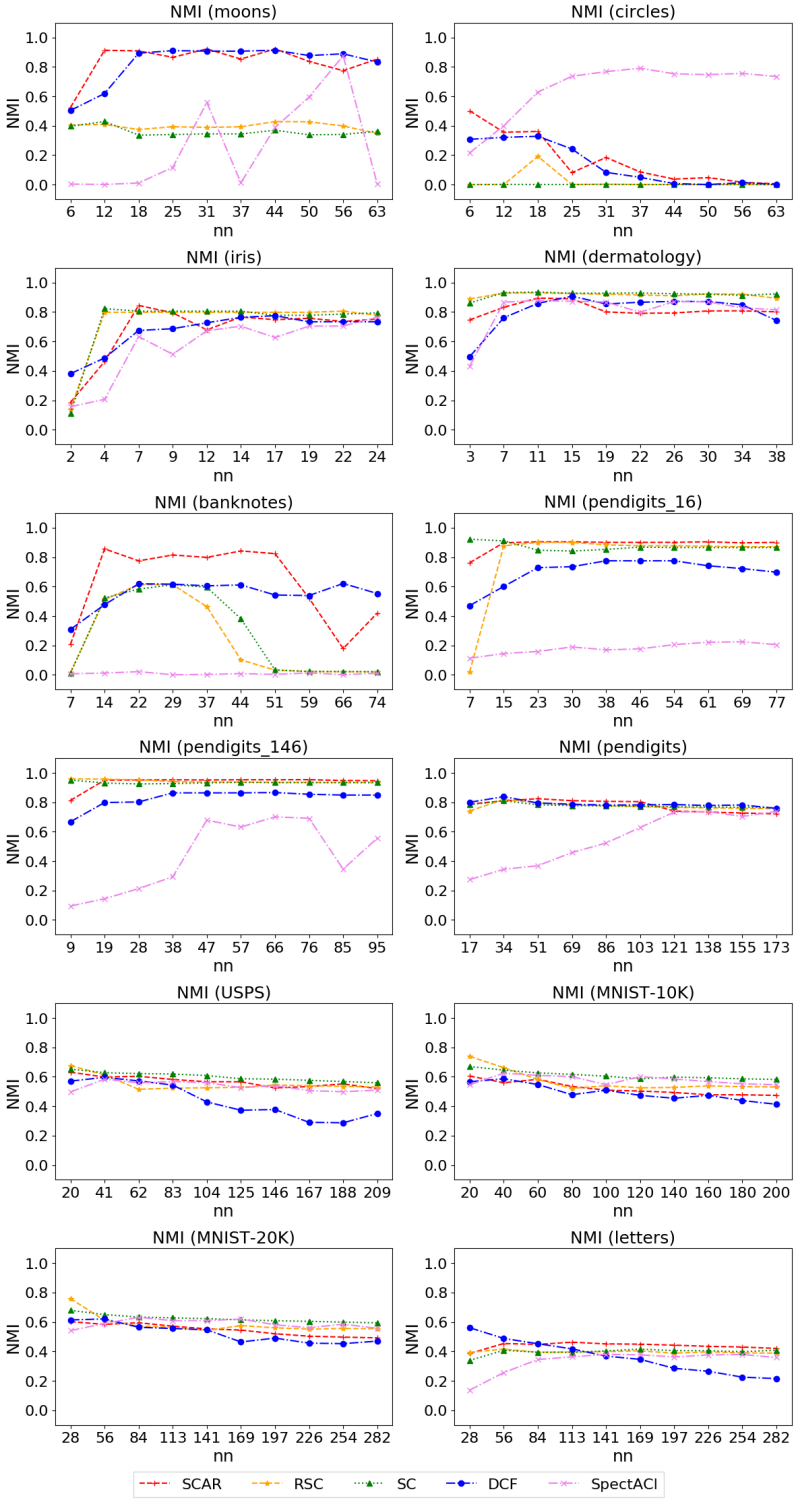


Figure 3.4: *Comparison of NMI of all methods with their best parameter settings on all datasets depending on nn.*

3.2.5.2.2 Robustness against Noise

To evaluate SCAR's robustness against noise, the parameter settings for *nn* and α are fixed, and only the amount of *jitter* is modified in the range of $[0.0, 0.05, 0.1, \ldots, 0.03]$ on the moons dataset. The left graph in Figure 3.5 shows that SCAR consistently outperforms other models on the moons dataset for high noise levels ($noise > 0.2$). The NMIs of most comparative methods drop heavily for noise values over 0.1, resp., 0.2. Qualitative results can also be seen in Figure 3.2, where SCAR is the only method able to correctly discern the two moons for a comparably high noise level of $noise = 0.15$. The right graph in Figure 3.5 gives all runtimes in seconds. SCAR shows an almost constant runtime over different levels for *noise*. For RSC, higher runtimes are observable as the eigendecomposition on the whole Laplacian is computed in each iteration. Notably, DCF also shows increased runtimes for low noise values due to higher densities within the clusters. The efficiency of SCAR evaluated on different benchmark datasets is further discussed in the next section.

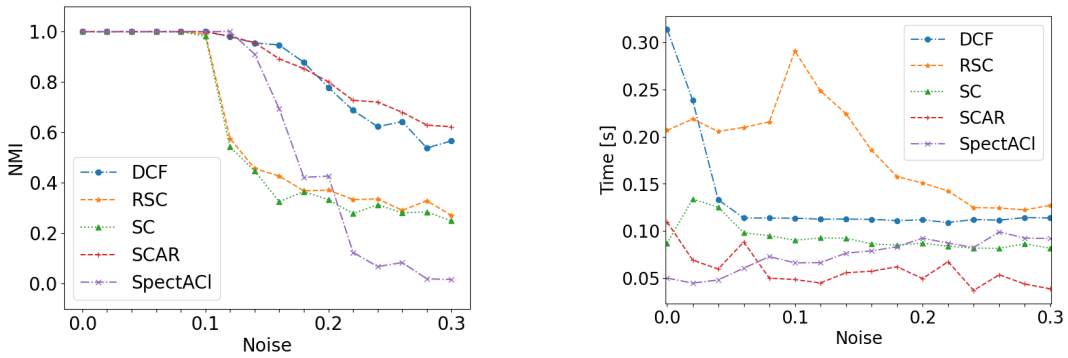
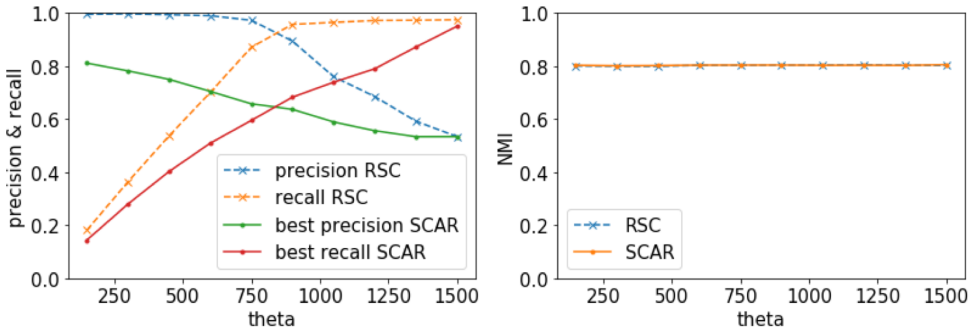


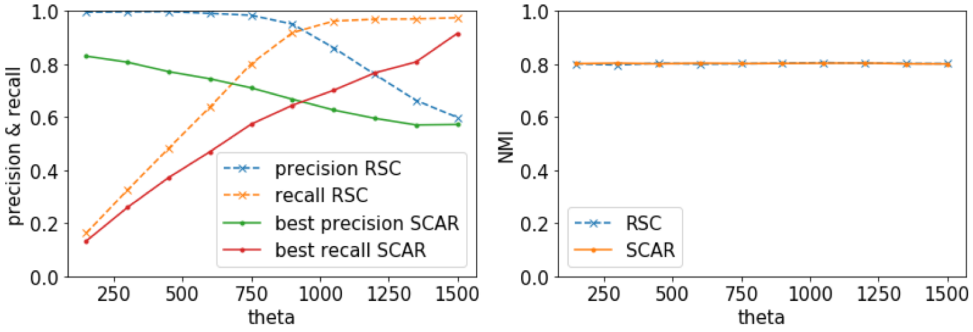
Figure 3.5: *NMI scores (left) and runtime in [s] (right) for noise* $\in [0, \ldots, 0.3]$ *in* 0.02 *steps on moons dataset.*

Similar to [20], the thesis also examines the robustness against *noisy edges*: Gaussian distributed clusters (blobs) are generated and versions of the moons datasets where "corrupted" edges are added to the associated *k*NN graph. I.e., edges between nodes of different clusters were added using the planted-partition model. Intra-cluster edges were created with a probability of 30% and noise edges were added such that 10%, resp., 20% of all edges in the *k*NN graph were corrupted. We evaluate the precision $p = |\mathscr{E}_c \cap \mathscr{E}_r| / |\mathscr{E}_c|$ and recall $r = |\mathscr{E}_c \cap \mathscr{E}_r| / |\mathscr{E}_r|$, where $\mathscr{E}_c$ is the set of corrupted edges and $\mathscr{E}_r$ is the set of edges removed by SCAR. In contrast to [20], the thesis also regards the effect of removing corrupted edges on the clustering quality. Figures 3.6a and 3.6b show precision and recall of the detected corrupted edges, as well as the achieved NMIs of RSC and SCAR for increasing θ. Even though precision and recall – implying the quality of detecting corrupted edges – of SCAR's results are lower than for RSC, this does not affect the overall clustering quality. Instead, the constant NMI scores, while increasing θ for both cases (10% and 20% added noise edges), indicate that corrupted edges do not affect the obtained clustering quality for Gaussian distributed clusters. Figures 3.6c and 3.6d imply that this

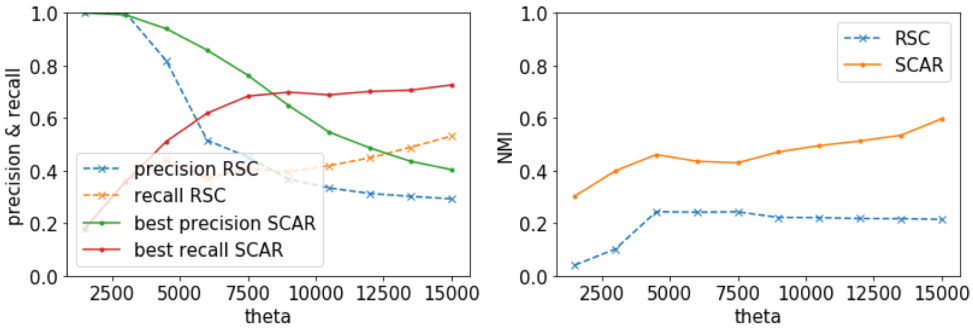
is different for moons datasets. Here, SCAR surpasses RSC w.r.t. precision and recall on both noise settings throughout almost all tested values for θ. In contrast to the Gaussian distributed clusters, for the moons dataset, removing corrupted edges enables a higher clustering quality: The NMIs of SCAR are significantly higher than the NMIs for RSC throughout all tested values for θ. Figure 3.6 shows that SCAR surpasses RSC in the detection of corrupted edges exactly where these corrupted edges impede a high quality clustering by connecting hard-to-distinguish clusters.



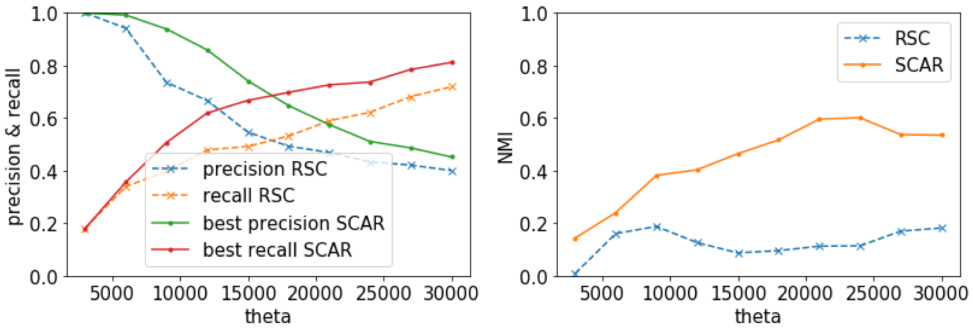
(a) 10% artificial noise edges added to blobs



(b) 20% artificial noise edges added to blobs



(c) 10% artificial noise edges added to moons



(d) 20% artificial noise edges added to moons

Figure 3.6: *Precision and recall (left) and NMI (right) for 10% or 20% artificial noise edges added to blobs (n=1000, k=20) resp. moons averaged over random_state=*$[0-9]$.

3.2.5.3 Runtime Analysis

In this section, the model's runtime is evaluated. In Section 3.2.5.3.1, a general overview of SCAR's efficiency compared to state-of-the-art methods is provided. The thesis shows scalability experiments in Section 3.2.5.3.2 and regards the complexity in Section 3.2.5.3.3

3.2.5.3.1 Efficiency

Table 3.2 shows on the right the minimum runtime in seconds for the trials with the highest NMI scores as well as the minimum runtime of all tested parameter settings in brackets. We observe that SCAR yields its best results w.r.t. the NMI almost always as the fastest or second fastest algorithm. SCAR can generally provide faster results than standard SC. The speed-up, in particular, increases with larger datasets. Only on the quite small dermatology dataset, SCAR runs 0.02 seconds longer than SC. The highest speed-up is reached on the letters dataset, where SCAR is more than 20 times faster than SC, while simultaneously increasing the NMI by 10%. Using a similar design and notion of noise as RSC, it is noteworthy that SCAR surpasses RSC w.r.t. the runtime on every tested dataset. Note also, that RSC already accelerates the eigendecomposition by leveraging IRLM. SCAR reaches a maximum acceleration factor of 6.4 in relation to RSC for the heavily noisy circles dataset, where it simultaneously improves the clustering quality from an NMI (ARI) of 0.19 (0.08) to 0.50 (0.57). For further runtime comparisons with RSC and SC, see Section 3.2.5.5.2. Even though DCF shows fast runtimes as well as good clustering results for most datasets, it cannot reach acceptable runtimes on high-dimensional datasets like USPS or MNIST variations (marked in red). Whereas experiments on lower-dimensional datasets show comparable runtimes in the same order of magnitude for all algorithms, DCF needs on these three high-dimensional datasets more than ten times longer than SCAR. Further investigations on the dependence of all algorithm's runtimes can be found in Section 3.2.5.3.2. SpectACl shows, similar to SCAR, good runtimes for its best results, but mostly returns significantly worse clustering results. Especially SpectACl's performance on the datasets banknote and the first two pendigits versions (marked in red) is of surprisingly low quality.

For some parameter settings, algorithms may have significantly lower runtimes than for others. E.g., for a small number of nearest neighbors *nn*, the respective nearest neighbors graph has less edges and, thus, most operations performed on it are faster. Analyzing the values in brackets in Table 3.2, we see the best runtimes over all tested parameter settings that can be reached for each experiment and each algorithm. I.e., in contrast to the runtimes regarded in the last paragraph, where parameter settings for a high NMI are optimized, we now optimize parameter settings for a low runtime. Also here, SCAR reaches most often the fastest runtimes. Notably, even for the most suitable parameter settings, DCF cannot achieve an acceptable runtime on high-dimensional datasets like USPS and MNIST variations (marked in red).

It can be observed that for all datasets, the minimum runtime for the best NMI results were usually close to the respective minimum runtime over all tested settings. More precisely, most

of them were at maximum twice as high as the fastest runtime for the respective algorithm. This supports the idea that the model has a stronger influence on its runtime than the selected parameter setting.

In conclusion, SCAR is stable in its anticipated clustering quality and yields good results at high speed.

3.2.5.3.2 Scalability on Synthetic Datasets

Figure 3.7 shows the scalability of SCAR on the moons and blobs datasets, with a fixed noise of 0.15 for moons and a default cluster standard deviation of 1.0 for blobs. On the former, the number of data points were scaled in the range $[100, \ldots, 50k]$. On the latter, the number of features $[2, \ldots, 50k]$ were scaled. The number of neighbors that are taken into account for the construction of the kNN graph is set to $nn = \sqrt{n}$ in both experiments, where n denotes the number of data samples. All other parameters are frozen. The left diagrams show the obtained NMI scores, and the right diagrams show the elapsed time for all evaluated methods. On the moons dataset, SCAR's scalability oupterforms RSC and SC w.r.t. both, computational time as well as obtained NMI scores for increasing sample size. For smaller datasets, the novel approach also shows superior performance compared to DCF, which cannot be maintained for increasing the sample-size. However, DCF's runtimes deteriorate for higher dimensionalities, as can be seen in the lower part of Figure 3.7 (note the log-scale). SCAR's runtime stays almost linear in the number of features. In Figure 3.8, DCF's unfortunate runtime behavior w.r.t. the dimensionality can also be observed on larger real-world datasets with higher dimensionality. While DCF yields low runtimes for large datasets if they are low-dimensional, e.g., for letters, its runtime tremendously increases on USPS, MNIST-10K and MNIST-20k.

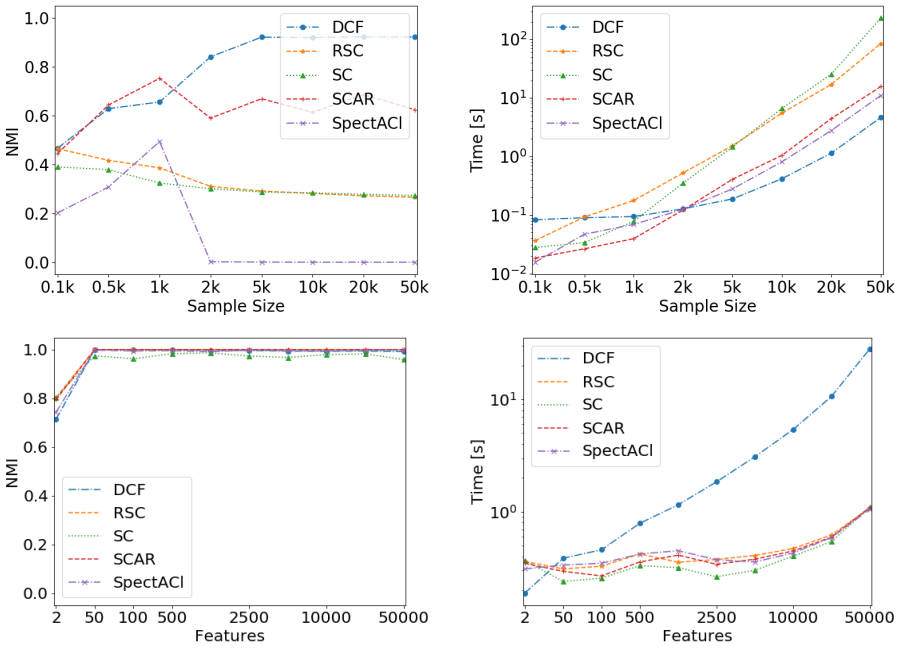


Figure 3.7: *NMI scores (left) and runtime in seconds (right). Top: Moons dataset (noise=0.15) with varying n. Bottom: Blobs dataset (n=1000, k=20, random_state=None) with varying d.*

3.2.5.3.3 Complexity Analysis

Having the same fundamental structure as RSC, the thesis refers for the complexity analysis on the explanations of [20], showing a runtime approximately linear in the number of edges. In the following, it is elaborated on the differences between SCAR and RSC that potentially influence the complexity (see also Figure 3.3). In Step (1), the weighted kNN graph is calculated in contrast to the unweighted kNN graph for RSC and a Gaussian kernel on the edge weights is applied. These changes do not increase the runtime complexity, as all edges of the kNN graph are accessed in both approaches. In Step (3), [20] use the power iteration for the eigendecomposition. For SCAR, the runtime is reduced by using the Nyström method, see Section 3.2.3.2. In Step (5), the rows of the approximated, cleansed matrix $\tilde{H} \in \mathbb{R}^{n \times k}$ are normalized, where $\tilde{H}$ contains the first k vectors that are needed for the subsequent clustering step. Usually, we have $k \ll \sqrt{n}$, such that the complexity is not increased when working on a kNN graph (containing approximately $\mathcal{O}(n\sqrt{n})$ edges [149]). Overall, SCAR reaches a similar complexity as RSC, which is approximately linear in the number of edges, while improving the runtime. The experiments in Section 3.2.5.3 confirm the improved runtime w.r.t. RSC.

3.2.5.4 Effectiveness and Efficiency

Figure 3.8 summarizes the models' performances on the various datasets, where the x-axis shows the runtime and the y-axis shows the clustering scores. Optimal results are located in the upper left reflecting a high NMI score reached within a short amount of time. It shows only the best runs of all methods to reduce visual clutter, i.e., only runs that yielded at least 75% of the best NMI score reached by the respective method are shown as single dots. On the highly noisy moons dataset, SCAR's robustness and efficiency dominates the other methods in terms of both, clustering performance and runtime. On small real-world datasets (iris, dermatology, banknotes and the pendigits variations), SCAR is highly competitive with other state-of-the-art models w.r.t. NMI and runtime. As all tested methods have runtimes below one second for all smaller datasets, larger datasets are more expressive for runtime analyses. Thus, in the following (as well as in Figure 3.10), the datasets with more than 5000 points (pendigits, USPS, MNIST versions and letters) are regarded when investigating runtimes. Noteworthy, SCAR is comparably fast on these datasets *and* reaches low runtimes with a comparably low variance. For the low-dimensional datasets pendigits and letters, DCF is even faster than SCAR, but for higher-dimensional datasets (USPS and MNIST versions) advantages of using any of the newer spectral clustering approaches become clear, as DCF's runtime does not scale with the dimensionality. In summary, Figure 3.8 demonstrates that SCAR nearly always outperforms its competitors in either runtime, clustering quality, or both and particularly highlights SCAR's reliability.

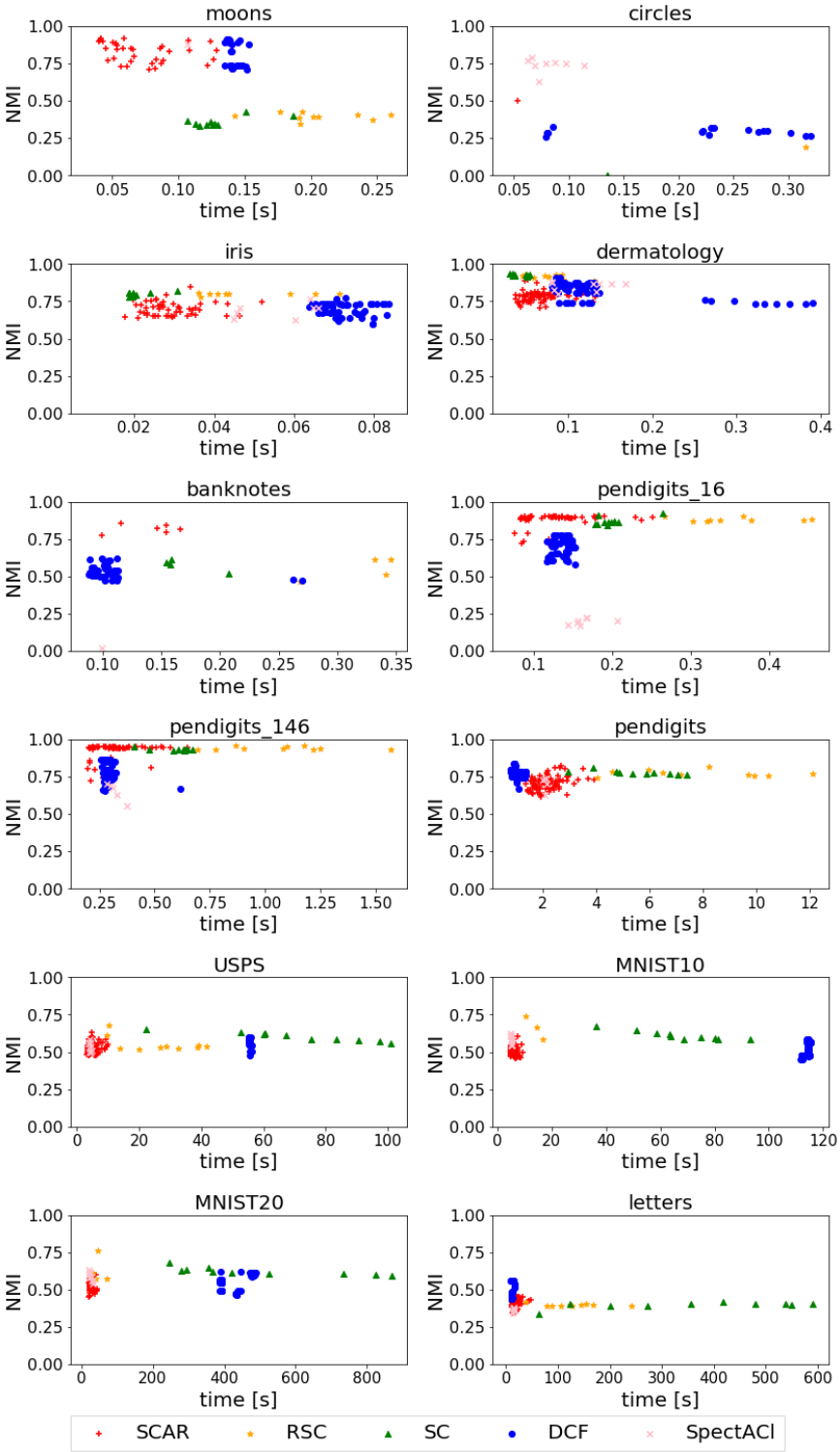


Figure 3.8: *Runtimes and NMIs of all experiments that reached at least 75% of each method's respective best NMI.*

3.2.5.5 Improvements over RSC and SC

In the following, the thesis examines the improvements of SCAR over RSC and the original Spectral Clustering algorithm (SC) in more detail. Section 3.2.5.5.1 regards the single components that differentiate SCAR from RSC as well as their functional interaction. Section 3.2.5.5.2 regards runtime improvements over RSC.

3.2.5.5.1 Effectiveness Improvements of SCAR over RSC and SC

Figure 3.9 shows NMIs on the highly noisy moons dataset for various settings for *nn* and different methods: on the left, RSC is compared with a straight-forward Nyström-accelerated version of RSC and SCAR. On the right, an ablation study is performed w.r.t. the changes between RSC and SCAR. (The results are condensed by setting α to the recommended default value $\alpha = 0.7$). The left part of the figure shows that a simple speed-up of RSC would lead to significantly worse results, whereas SCAR drastically improves the results of RSC. On the right, we can see that each of SCAR's components is chosen meaningfully, leading to an improvement of quality that is reached by the elaborated combination of concepts rather than any single adaption. The reasons for the individual components are regarded in Section 3.2.4 and their impacts and synergies are explained in the following. Using an unweighted graph can deliver good results on the moons dataset, if exactly the right number for *nn* is chosen (i.e., such that only very few corrupted edges exist). However, as seen in the first line of Figure 3.9 on the right, this leads to a strong and unpredictable dependence on guessing a good value for *nn*. Weighting the edges also allows for a more meaningful sampling of the edges for the Nyström method with the adjusted sampling method being applied: as corrupted edges connect nodes of different clusters (and distances between clusters are larger than distances inside clusters) they tend to be longer than non-corrupted edges. Applying the Gaussian kernel for calculating the edges' weights, this leads to smaller degrees of nodes connected by corrupted edges. Thus, sampling the nodes with higher degrees allows to sort out corrupted edges (compare with line 3 on the right of Figure 3.9). Using an unnormalized Laplacian further enhances the distinguishability between corrupted and non-corrupted edges [85] in the eigenspace, reinforcing the positive effects of the adjusted sampling method heavily (see line 2 on the right of Figure 3.9). Small perturbances in the data can be compensated by normalizing the rows [59]. That accounts for jitter and pushes points of a cluster even closer in eigenspace, which robustifies the adapted sampling step and further improves the clustering (line 4 on the right of Figure 3.9).

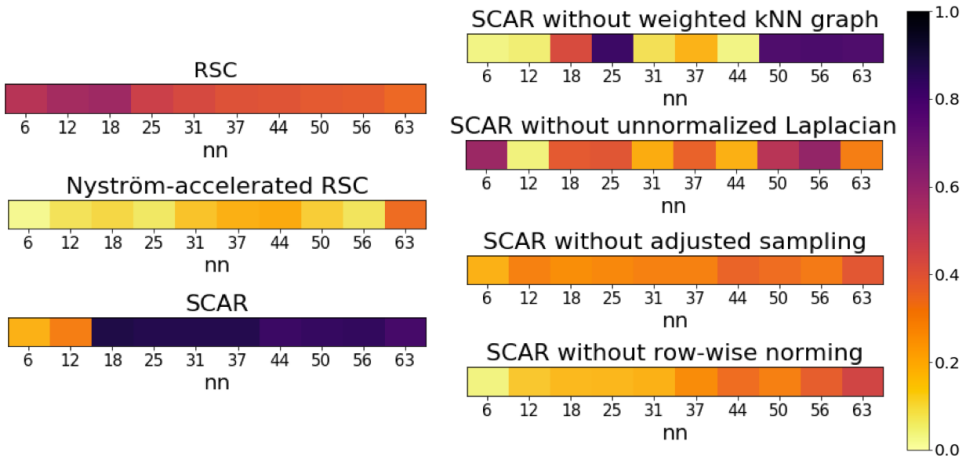


Figure 3.9: *Ablation study of SCAR's NMI performance on moons (avg. over 10 random instantiations) depending on nn and for fixed $\alpha = 0.7$. Dark colors imply better NMI scores.*

3.2.5.5.2 Efficiency Improvements over RSC and SC

Figure 3.10 shows SCAR's substantial runtime accelerations over RSC depending on their parameter settings on four larger benchmark datasets (where $n \geq 5000$, resp., $runtime > 1s$). In Figure 3.8, runtimes and their respective NMIs are shown for all methods. In particular for larger datasets, SCAR nearly always outperforms SC and RSC regarding runtime significantly. A more thorough discussion on the proper choice of hyperparameters α and *nn* is given in Section 3.2.5.6.1.

3.2.5.6 Hyperparameter Tuning

In this section, the thesis examines the influence of various parameter settings on the model's performance. In Section 3.2.5.6.1, the thesis examines the portion α denoting the number of landmarks chosen for the Nyström subsample and the number of nearest neighbors *nn* for the construction of the *k*NN graph. In Section 3.2.5.6.2, the performance of various decomposition methods on the sampled submatrix is evaluated and how the clustering quality and computational time depends on different configurations. In Section 3.2.5.6.3, the influence of the parameter θ on the models' performances of SCAR and RSC is investigated.

3.2.5.6.1 Number of Landmarks and Number of Neighbors

In Figure 3.11, the NMI scores for all tested datasets depending on *nn* and α are shown, where darker, resp., lighter colors reflect higher, resp., lower NMI scores. For a more thorough analysis of the impact of the number of neighbors, $2\sqrt{n}$ is used as an upper bound for *nn* [46]. We see that the choice for *nn* and α has a strong effect on the clustering quality: The quality of smaller datasets depends more heavily on a proper choice of α compared to larger datasets. The experiments show that a higher amount of landmarks improves clustering results. Furthermore,

the illustration reveals that on larger datasets, the performance is improved whenever the *k*NN graph retains its sparse nature, i.e., by lowering the amount of *nn*. This effect also heavily improves the efficiency of the proposed method as discussed in Section 3.2.5.3.1. On USPS, as well as on the MNIST datasets, we observe higher peaks for lower values of *nn*. On smaller datasets, it is more likely that the *k*NN graph connects samples from distinctive cluster, i.e., the graph contains misleading information. Comparing iris and dermatology, the evaluation revealed that for the latter, it is more favorable to choose a smaller *nn* to identify the six clusters properly, whereas on iris, with three clusters, we can choose higher values without mingling the information of separate clusters. Per default, the suggestion is to use values $\alpha = 0.7$ and $nn = \sqrt{n}$.

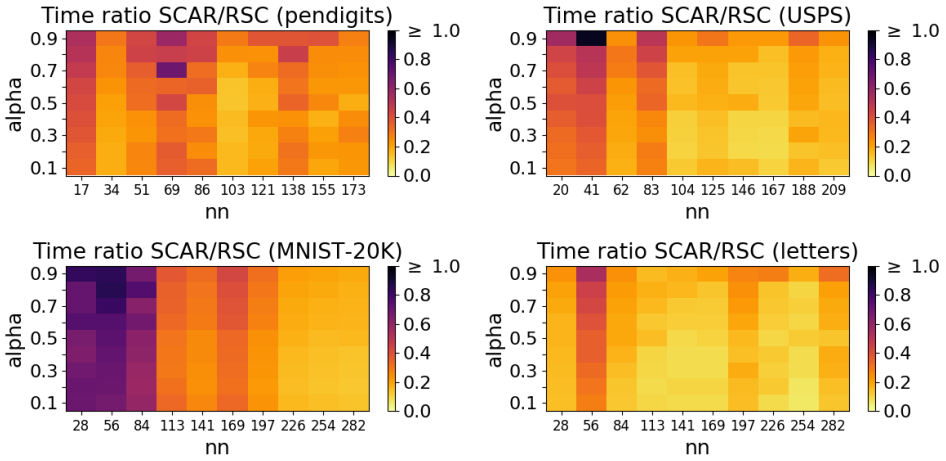


Figure 3.10: *Summary of time ratios SCAR / RSC depending on nn and α for large real world datasets. Lighter colors imply better (i.e., faster) results for SCAR.*

While good clustering results are a prerequisite for useful clustering algorithms, SCAR's major benefit is its runtime acceleration. Figure 3.10 summarizes obtained runtime quotients of SCAR compared to RSC for four large real-world datasets and their dependence on *nn* and α. It only displays the larger datasets here, as they require runtimes for RSC $\gg 1s$ (see Table 3.2), and, therefore, an acceleration analysis is more meaningful. The values in each heatmap depict the ratio of runtimes between SCAR and RSC, i.e., $runtime(SCAR)/runtime(RSC)$. Consequently, smaller values indicate faster runtimes of SCAR compared to RSC. The effect and strength of the Nyström method can be observed for all larger datasets. By sampling only a submatrix in order to approximate the spectrum as a whole, we observe a performance boost compared to RSC. The impact of the choice of α is shown on the y-axis, whereas the effect of *nn* is shown on the x-axis. The experiments show that SCAR has significantly lower runtimes than RSC even for high values of α, further supporting the quite high recommended choice for $\alpha = 0.7$. For larger values of *nn* SCAR's speed-up becomes even clearer: Larger *nn* lead to more edges in

the *k*NN graph and, therefore, more acceleration potential for SCAR over RSC as the graph is more dense. Figure 3.4 indicates that SCAR's clustering results are relatively robust against the choice of *nn*. Thus, SCAR's runtime improvements over RSC do not have a negative effect on its clustering performance.

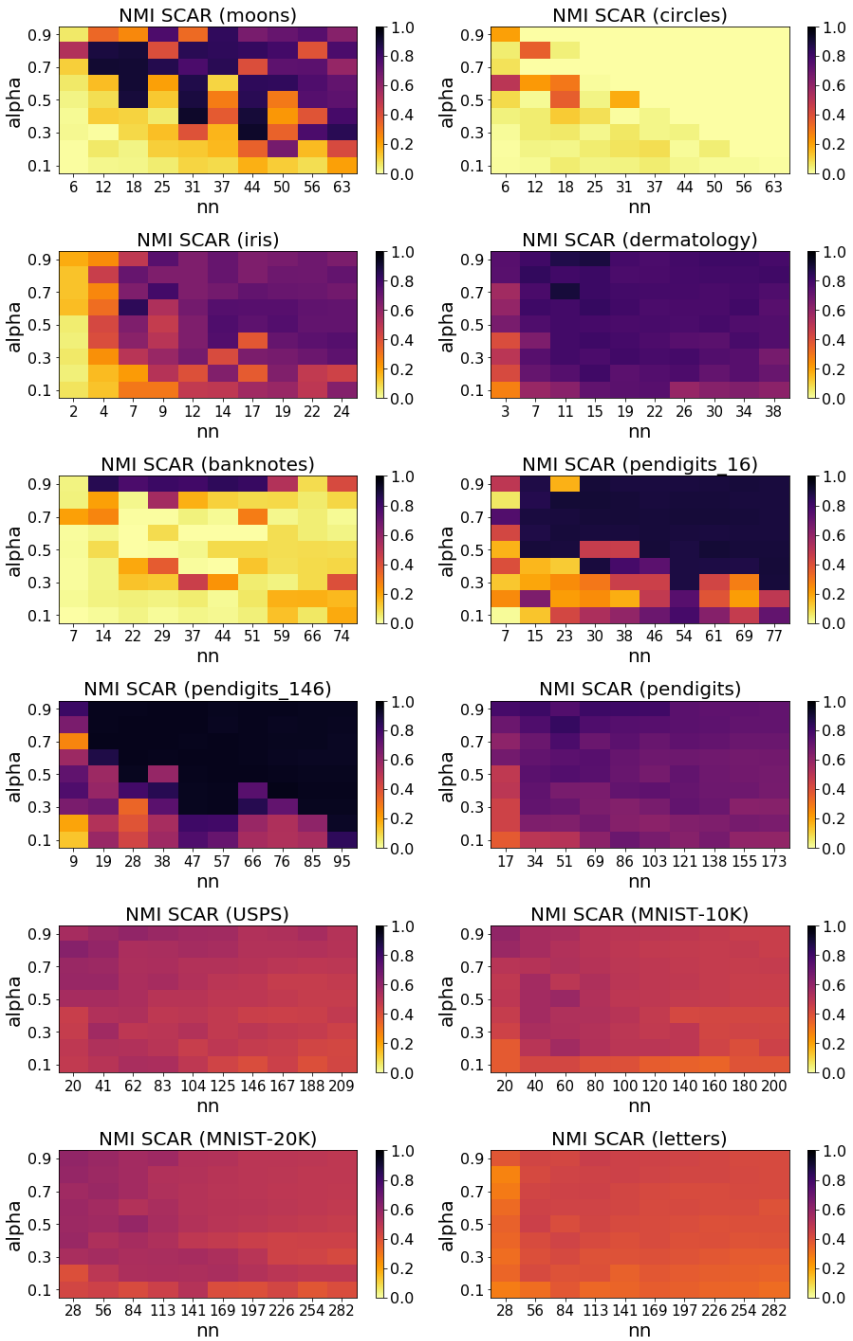
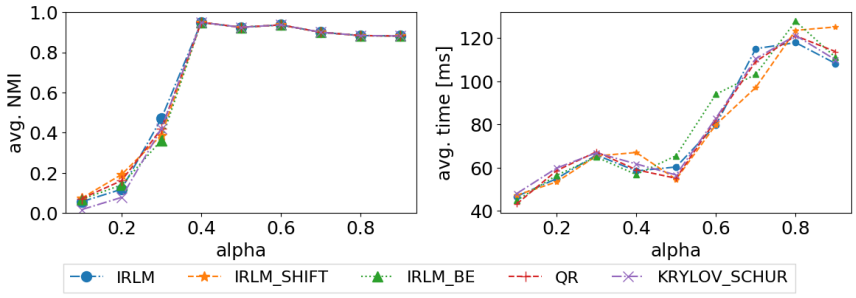


Figure 3.11: *Summary of NMI obtained with SCAR depending on nn and α for all datasets.*

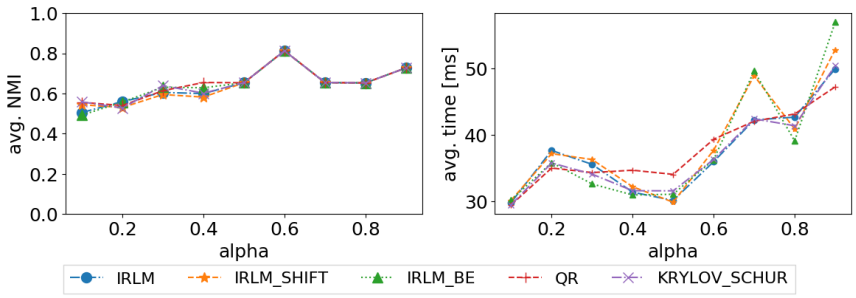
3.2.5.6.2 Decomposition of Submatrix

In the following, the thesis evaluates commonly used decomposition methods on the sampled submatrix of the Nyström Approximation explained in Section 3.2.3.2. Figure 3.12 shows the highest observed NMI scores (left) within 10 trials as well as the respective runtimes (right) with a fixed value of *nn* for each dataset. As the submatrix in the Nyström method is symmetric, the *Implicitly Restarted Lanczos Method* (IRLM) is applied which is based on power iterations and has also been used in [20] as decomposition heuristic. Additionally, various variants of IRLM are evaluated with *-Shift* applying a shift-inversion on the spectrum to transform the smallest eigenvalues to be the highest, and *-BE* for which eigenvalues are approximated from both ends of the spectrum. For the latter, [156] showed, that approximating eigenpairs from both ends of the spectrum can speed-up the convergence. A standard QR decomposition is also evaluated, as well as the *Krylov-Schur* decomposition as proposed by [195].[20] Empirically, all decomposition methods yielded similar qualitative results w.r.t. the NMI score. Examining the runtimes on smaller datasets, we observe a slight overhead in the computation of the shifting operation for *IRLM-Shift*, as well as in applying a sampling from both ends of the spectrum. On larger datasets, this effect flattens out and the *Krylov-Schur* decomposition that is optimized towards large, sparse matrices shows a marginal benefit for larger α values. In the experiments, the standard IRLM is used as default heuristic for the computation of the eigenpairs as it showed competitive results over the full range of the tested datasets.

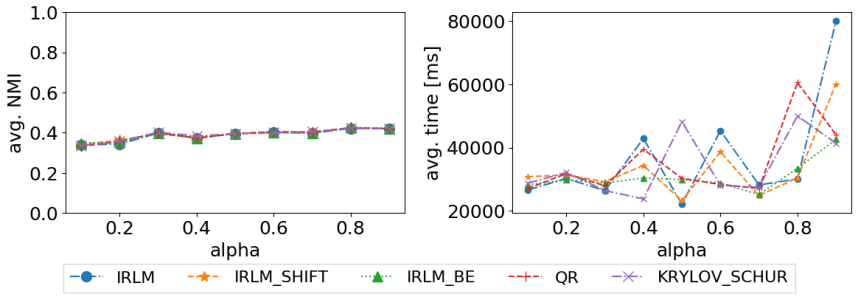
[20] State-of-the-art libraries are used, where IRLM and its variants are implemented in ARPACK, QR in LAPACK, and krylov-schur as part of SLEPc/PETSc



(a) *synthetic data - moon*



(b) *real data - Iris*



(c) *real data - Letter Recognition*

Figure 3.12: *Avg. NMI scores (left) and runtimes (right) for decomposition methods IRLM, IRLM-Shift, IRLM-BE, QR and krylov-schur on different datasets.*

3.2.5.6.3 Influence of Parameter θ

In Figure 3.13, the influence of parameter θ is evaluated on the clustering's quality and runtimes for SCAR and RSC [20]. As argued in Section 3.2.5.6.1, the parameter nn is fixed to $nn = \sqrt{n}$. The number of expected corruptions is scaled in the dataset logarithmically: $\theta \in [10,\ 100,\ 1k,\ 10k]$. On the moons dataset, the proposed approach outperforms RSC almost over the full range of chosen θ whilst drastically reducing the computational time as shown on the right. Generally, increasing the sparsity threshold might lead to a clearer separation, however, the clustering quality suffers for very large values as clean edges might be attached to the corrupted graph A^c.

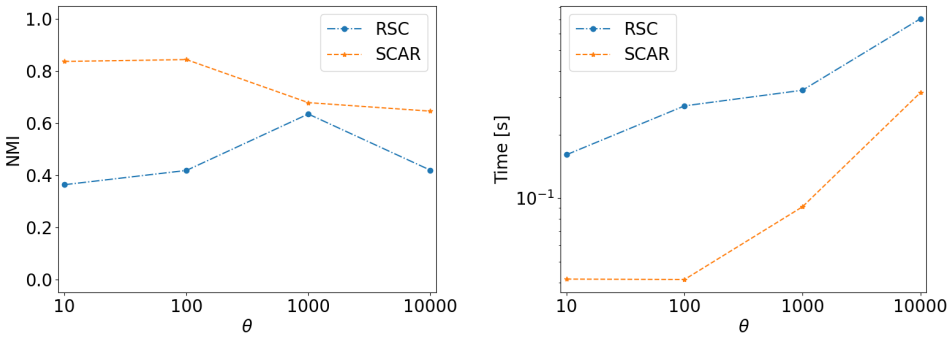


Figure 3.13: *NMI scores (left) and runtime in [s] (right) for* $\theta \in [10, 100, 1k, 10k]$ *on moons dataset (noise*=0.15*).*

3.3 Summary

The thesis introduced SCAR, a novel robust and efficient clustering method. It elucidates the benefits from Robust Spectral Clustering [20] enhanced by the Nyström method for an accelerated computation of the eigendecomposition. The sensitivity to noisy input data as well as the runtime complexity compared to standard Spectral Clustering are significantly reduced.

In a thorough experimental study, the thesis compares SCAR's clustering quality with state-of-the-art models showing highly competitive results on real-world benchmark datasets, as well as its robustness against noise on artificial data. Robustness w.r.t. noisy edges in the similarity graph of the data as well as robustness w.r.t. jitter in the original data are evaluated, tackling the two most difficult types of noise for clustering. SCAR consistently yielded low runtimes, in particular it is significantly faster than RSC and SC, while returning highly competitive clustering qualities on real-world and synthetic data. SCAR is recommendable when looking for a reliable, fast and robust clustering method on large and high-dimensional datasets that tend to be noisy.

4

Attributed Graphs

In nature we never see anything isolated, but everything in connection with something else which is before it, beside it, under it and over it.

Johann Wolfgang von Goethe
1749-1832

Attribution

This Chapter uses material from the following publication:

- Christian M. M. Frey, Yunpu Ma, and Matthias Schubert. Sea: Graph shell attention in graph neural networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022. [65]

See § 1.4 for an overview of incorporated publications and the author's attribution.

Highlights

- Integration of expert-routing into Graph Neural Networks (GNNs);
- Novel Graph Shell Attention (SEA) models capturing short- and long-term dependencies, simultaneously;
- Experiments showing a reduction in the number of model parameters to be learned compared to state-of-the-art models;

4.1 Deep Learning on Attributed Graphs

In Deep Learning, we observe that *Convolutional Neural Networks* (CNNs) are well-defined over grid-structured inputs like images, while *Recurrent Neural Networks* (RNNs), *Long-Short Term Memories* and *Gated Recurrent Units* (GRU) are suitable for processing sequences (e.g., text, signals). In order to process graph data with deep learning heuristics, it was necessary to define a new kind of deep learning architecture. A general framework for defining neural networks on graph data is based on the so-called *message-passing paradigm*. Similarly, the idea is to generate nodes' representations that reflect the graph's structure and additional information being provided by the input data. Generally, this is achieved by neural message passing, where the vector representations of nodes are exchanged and updated using neural networks [73].

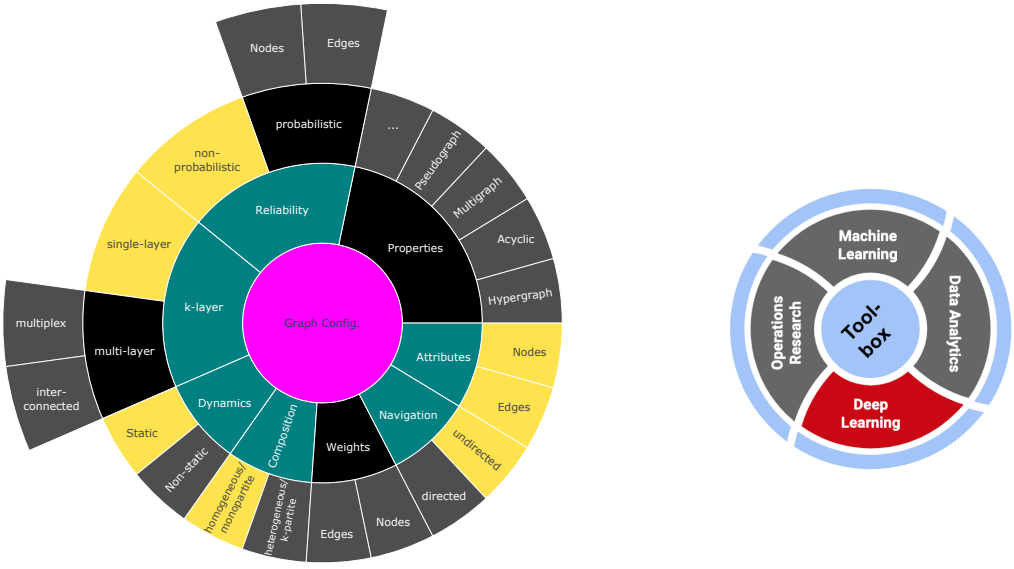
§ 4.2 covers learning graph data representations using a novel heuristic based on deep learning techniques. The key idea is to use a transformer-based attention mechanism in combination with a routing mechanism steering the process of learning the nodes' representations to a set of experts. The experts differ in processing nodes in the neighborhood of a query node. This novel heuristic is referred to as *Graph Shell Attention* (SEA).

4.2 Graph Shell Attention in Graph Neural Networks

A common problem in *Graph Neural Networks* (GNNs) is known as *over-smoothing*. By increasing the number of iterations within the message-passing of GNNs, the nodes' representations of the input graph align and become indiscernible. The latest models employing attention mechanisms with *Graph Transformer Layers* (GTLs) are still restricted to the layer-wise computational workflow of a GNN that are not beyond preventing such effects. In this thesis, the GNN architecture is relaxed by means of implementing a routing heuristic. Specifically, the nodes' representations are routed to dedicated experts. Each expert calculates the representations according to their respective GNN workflow. The definitions of distinguishable GNNs result from k-localized views starting from the central node. This procedure is called *Graph **S**he**l**l **A**ttention* (SEA), where experts process different subgraphs in a transformer-motivated fashion. Intuitively, by increasing the number of experts, the models gain in expressiveness such that a node's representation is solely based on nodes that are located within the receptive field of an expert. The novel architecture is evaluated on various benchmark datasets showing competitive results while drastically reducing the number of parameters compared to state-of-the-art models.

Graph Configuration.
The configuration for the novel graph neural network model is shown in Figure 4.1a. The setting is a single-layer, static, non-probabilistic graph. The graphs used for the evaluation contain attributes on the nodes, and two benchmark datasets also contain attributes on the edges. These

additional nodes', respectively, edges' attributes point to the vector representations in the embedding space. The technical domain of the proposed model is *Deep Learning*.



(a) *Graph Configurator* ***(b)*** *Toolbox*

Figure 4.1: *Graph Configuration and methodology's classification used to solve graph shell attention (SEA) in graph neural networks (GNN)*

4.2.1 Motivation

Graph Neural Networks (GNNs) have been proven to be an important tool in a variety of real-world applications building on top of graph data [224]. These range from predictions in social networks over property predictions in molecular graph structures to content recommendations in online platforms. From a machine learning perspective, we can categorize them into various theoretical problems that are known as *node classification*, *graph classification/regression* - encompassing binary decisions or modeling a continuous-valued function -, and *relation prediction*. This thesis proposes a novel framework and show its applicability on graph-level classification and regression, as well as on node-level classification tasks.

The high-level intuition behind GNNs is that by increasing the number of iterations $l = 1, \ldots, L$, a node's representation contains and, therefore, relies successively more on its k-hop neighborhood. However, a well-known issue with the vanilla GNN architecture refers to a problem called *over-smoothing* [226]. In simple words, the information flow in GNNs between two nodes $u, v \in \mathcal{V}$, where $\mathcal{V}$ denotes a set of nodes, is proportional to the reachability of node v on a k-step random walk starting from u. By increasing the layers within the GNN architecture, the

information flow of every node approaches the stationary distribution of random walks over the graph [89]. As a consequence, the localized information flow is getting lost, i.e., increasing the number of iterations within the message-passing of GNN results in representations for all the nodes in the input graph that align and become indiscernible [153]. One strategy for increasing a GNN's effectiveness is adding an attention mechanism. An adaption of the *Transformer* model [206] on graph data has been introduced as *Graph Transformer Layer* (GTL) [49]. Generally, multi-headed attention shows competitive results whenever we have prior knowledge to indicate that some neighbors might be more informative than others. The novel framework proposed in the following chapters further improves the representational capacity by adding an expert heuristic into the GTL architecture. More specifically, to compute a node's representation, a routing module first decides upon an expert that is responsible for a node's computation. The experts differ in how their k-hop localized neighborhood is processed and they capture individually various depths of GNNs/GTLs. We refer to different substructures that experts process as *Graph Shells*. As each expert attends to a specific subgraph of the input graph, the thesis introduces the concept of *Graph **Sh**ell **A**ttention* (SEA). Hence, whereas a vanilla GNN might suffer from oversmoothing the nodes' representations, the thesis introduces additional degrees of freedom in the new architecture to simultaneously capture short- and long-term dependencies being processed by respective experts.

4.2.2 Related Work

In recent years, the AI community proposed various forms of (self-)attention mechanisms in numerous domains. *Attention* itself refers to a mechanism in neural networks where a model learns to make predictions by selectively attending to a given set of data. The success of applying attention heuristics was further boosted by introducing the *Transformer* model [206]. It relies on scaled dot-product attention, i.e., given a query matrix Q, a key matrix K, and a value matrix V, the output is a weighted sum of the value vectors, where the dot-product of the query with corresponding keys determines the weight that is assigned to each value.

Transformer architectures have also been successfully applied to graph data. The work by Dwivedi et al. [49] evaluates transformer-based GNNs. They conclude that the attention mechanism in Transformers applied on graph data should only aggregate the information from local neighborhoods, ensuring graph sparsity. As in *Natural Language Processing* (NLP), where a positional encoding is applied, they propose to use Laplacian eigenvectors as the positional encodings for further improvements. In their results, they outperform baseline GNNs on the graph representation task. A similar work [117] proposes a full Laplacian spectrum to learn the position of each node within a graph. Yun et al. [230] proposed *Graph Transformer Networks* (GTN) that are capable of learning on heterogeneous graphs. The target is to transform a given heterogeneous input graph into a meta-path-based graph and apply a convolution operation afterwards. Hence, the focus of their attention framework is on interpreting generated meta-paths.

Another transformer-based architecture that has been introduced by Hu et al. [92] is *Heterogeneous Graph Transformer* (HGT). Notably, their architecture can capture graph dynamics w.r.t. the information flow in heterogeneous graphs. Specifically, they take the relative temporal positional encoding into account based on differences of temporal information given for the central node and the message-passing nodes. By including the temporal information, Zhou et al. [232] built a transformer-based generative model for generating temporal graphs by directly learning from the dynamic information in networks. The work of Ngyuen et al. [152] proposes another idea for positional encoding. The authors of this work introduced a graph transformer for arbitrary homogeneous graphs with a coordinate embedding-based positional encoding scheme. In [228], the authors introduced a transformer motivated architecture where various encodings are aggregated to compute the hidden representations. They propose graph structural encodings subsuming a spatial encoding, an edge encoding, and a centrality encoding. Furthermore, a work exploring the effectiveness of large-scale pre-trained GNN models is proposed by the *GROVER* model [172]. The authors include an additional GNN operating in the attention sublayer to produce vectors for Q, K, and V. Moreover, they apply single long-range residual connections and two branches of feedforward networks to produce node and edge representations separately. In a self-supervised fashion, they first pre-train their model on 10 million unlabeled molecules before using the resulting node representations in downstream tasks. Typically, all the models are built in a way such that the same parameters are used for all inputs. To gain more expressiveness, the motivation of the mixture of experts (MoE) heuristic [190] is to apply different parameters w.r.t. the input data. Recently, Google proposed *Switch Transformer* [57], enabling training above a trillion parameter networks but keeping the computational cost in the inference step constant. The thesis provides an approach how a similar routing mechanism can be integrated in GNNs.

4.2.3 Preliminaries

This section provides the basic notation (§ 4.2.3.1) being relevant for this chapter. Moreover, it provides concise recaps of *Graph Neural Networks* (§ 4.2.3.2), the general idea of *Transfomers* (§ 4.2.3.3), and *Graph Transformer Layer* (§ 4.2.3.4).

4.2.3.1 Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ denotes a set of nodes and $\mathcal{E}$ denotes a set of edges connecting nodes. Let $N_k(u)$ be the k-hop neighborhood of a node $u \in \mathcal{V}$, i.e., $N_k(u) = \{v \in \mathcal{V} : d_{\mathcal{G}}(u,v) \leq k\}$, where $d_{\mathcal{G}}(u,v)$ denotes the hop-distance between u and v on $\mathcal{G}$. For $N_1(u)$ we can simply write $N(u)$ and omit the index k. The induced subgraph by including the k-hop neighbors starting from node u is denoted by $\mathcal{G}_u^k$. Moreover, in the following, a real-valued representation vector $h_u \in \mathbb{R}^d$ is used for a node u, where d denotes the embedding dimensionality.

4.2.3.2 Recap: Graph Neural Networks

Given a graph $\mathcal{G}$ with node attributes $X_{\mathcal{V}} = \{X_u | u \in \mathcal{V}\}$ and edge attributes $X_{\mathcal{E}} = \{X_{uv} | (u,v) \in \mathcal{E}\}$ (cf. Definition 10), a GNN aims to learn an embedding vector h_u for each node u, and a vector $h_{\mathcal{G}}$ for the entire graph $\mathcal{G}$. The l-th layer is defined as:

$$m^l_{N(u)} = \text{AGGREGATE}^l(\{(h^l_v) : v \in N(u)\}) \tag{4.1}$$

$$h^{l+1}_u = \text{UPDATE}^l(h^l_u, m^l_{N(u)}), \tag{4.2}$$

where $N(u)$ is the 1-hop neighborhood set of u, h^l_u denotes the representation of node u at the l-th layer, and h^0_u is initialized as the node attribute X_u. From a spectral perspective, earlier rounds contain higher-frequency components than outputs from later rounds. Since h_u summarizes the information of central node u, it is also referred as *patch embedding* in the literature. A graph's embedding $h_{\mathcal{G}}$ is derived by a permutation-invariant readout function:

$$h_{\mathcal{G}} = \text{READOUT}(\{h_u | u \in \mathcal{V}\}) \tag{4.3}$$

A common heuristic for the readout function is to choose a function $\text{READOUT}(\cdot) \in \{mean(\cdot), sum(\cdot), max(\cdot)\}$.

4.2.3.3 Recap: Transformer

The vanilla Transformer architecture as proposed by Vaswani et al. [206] was originally introduced in the scope of *Natural Language Processing* (NLP) and consists of a composition of *Transformer layers*. Each Transformer layer has two parts: a self-attention module and a position-wise feedforward network (FFN). Let $H = [h^T_1, \ldots, h^T_n]^T \in \mathbb{R}^{n \times d}$ denote the input of the self-attention module where d is the hidden dimension and $h_i \in \mathbb{R}^{1 \times d}$ is the hidden representation at position i of an input sequence. The input H is projected by three matrices $W_Q \in \mathbb{R}^{d \times d_Q}$, $W_K \in \mathbb{R}^{d \times d_K}$, and $W_V \in \mathbb{R}^{d \times d_V}$ to get the corresponding representation Q, K, V. The self-attention is calculated as:

$$\begin{aligned} Q = HW_Q \qquad K = HW_K \qquad V = HW_V, \\ A = \frac{QK^T}{\sqrt{d_k}}, \quad Attn(H) = \text{softmax}(A)V \end{aligned} \tag{4.4}$$

Notably, in NLP as well as in computer vision tasks, the usage of transformer models were boosters behind a large number of state-of-the-art systems. Recently, the Transformer architecture has also been modified to be applicable to graph data [49], which is briefly recapped in the next section.

4.2.3.4 Recap: Graph Transformer Layer

As formalized in [49], a *Graph Transformer Layer* (GTL) update for layer $l \in [1..L]$ including edge features is defined as:

$$\hat{h}_u^{l+1} = O_h^l \Big\|_{i=1}^{H} \Big(\sum_{v \in N(u)} w_{uv}^{i,l} V^{i,l} h_v^l \Big), \tag{4.5}$$

$$\hat{e}_{uv}^{l+1} = O_e^l \Big\|_{i=1}^{H} (\hat{w}_{uv}^{i,l}), \text{where}, \tag{4.6}$$

$$w_{uv}^{i,l} = \text{softmax}_v(\hat{w}_{uv}^{i,l}), \tag{4.7}$$

$$\hat{w}_{uv}^{i,l} = \Big(\frac{Q^{i,l} h_u^l \cdot K^{i,l} h_v^l}{\sqrt{d_i}}\Big) \cdot E^{i,l} e_{uv}^l, \tag{4.8}$$

where $Q^{i,l}, K^{i,l}, V^{i,l}, E^{i,l} \in \mathbb{R}^{d_i \times d}$, and $O_h^l, O_e^l \in \mathbb{R}^{d \times d}$. The operator $||$ denotes the concatenation of attention heads $i = 1, \ldots, H$. Subsequently, the outputs $\hat{h}_u^{l+1}$ and $\hat{e}_{uv}^{l+1}$ are passed to feed-forward networks and succeeded by residual connections and normalization layers yielding the representations h_u^{l+1} and e_{uv}^{l+1}.

A graph's embedding $h_{\mathscr{G}}$ is derived by a permutation-invariant readout function w.r.t. the nodes in $\mathscr{G}$:

$$h_{\mathscr{G}} = readout(\{h_u | u \in \mathscr{V}\}) \tag{4.9}$$

A common heuristic for the readout function is to choose a function $\text{READOUT}(\cdot) \in \{mean(\cdot), sum(\cdot), max(\cdot)\}$.

4.2.4 Methodology

This section introduces the novel *Graph **Sh**ell **A**ttention* (SEA) architecture for graph data. SEA builds on top of the *message-passing paradigm* of *Graph Neural Networks* (GNNs) while integrating an expert heuristic.

4.2.4.1 Graph Shells Models

In the novel approach, *Graph Transformer Layers* (GTLs) [49] are implemented and the framework is extended by a set of *experts*. A routing layer decides which expert is most relevant for computing a node's representation. An expert's calculation for a node representation differs in how k-hop neighbors are stored and processed within GTLs.

Generally, starting from a central node, *Graph Shells* refer to subgraphs that include only nodes that have at maximum a k-hop distance (k-neighborhood). Formally, the i-th expert comprises the information given in the i-th neighborhood $N_i(u) = \{v \in \mathscr{V} : d_{\mathscr{G}}(u,v) \leq i)\}$, where

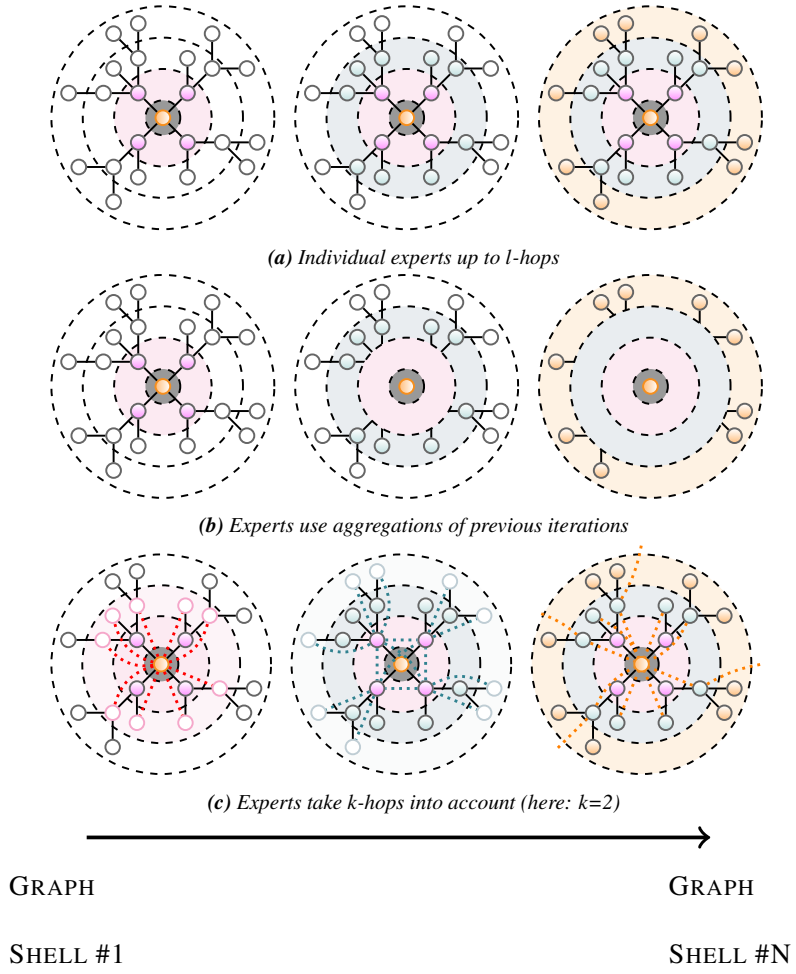


Figure 4.2: *Three variants of SEA models; for each model, the respective fields of* 3 *experts are shown from left to right.*

$u \in \mathcal{V}$ denotes the central node. The subgraph $\mathcal{G}_u^i$ is referred to as the expert's *receptive field*. Notably, increasing the number of iterations within GTLs/GNNs correlates with the number of experts being used. In the following, the thesis introduces three variants on how experts process graph shells:

• **SEA-GTL.** The first graph shell model exploits the vanilla architecture of GTLs for which shells are defined by the standard graph neural net construction. For a maximal number of L iterations, a set $\{E_i(u)\}_{i=1}^N$ of $N = L$ experts is defined. The embeddings after the l-th iteration are fed to the l-th expert, i.e., according to Equation 4.5, the information of nodes in $\mathcal{G}_u^l$ for a central node u has been processed. Figure 4.2a illustrates this model. From left to right, the information of nodes being reachable by more hops is processed. Experts processing information in early iterations refer to short-term dependencies, whereas experts processing more hops yield information of long-term dependencies.

• **SEA-AGGREGATED.** For the computation of the hidden representation h_u^{l+1} for node u on layer $l+1$, the second model employs an aggregated value from the preceding iteration. Following Equation 4.5, the aggregation function (sum) in GLT considers all 1-hop neighbors $N_1(u)$. For *SEA*-AGGREGATED, the aggregated value is propagated back to all of u's 1-hop neighbors. For a node $v \in N_1(u)$, the values received by v are processed according to an aggregation function $\text{AGG} \in \{mean(\cdot), sum(\cdot), max(\cdot)\}$. Formally:

$$h_u^{l+1} = \text{AGG}^l(\{h_v^{l+1} : v \in N(u)\}) \tag{4.10}$$

Figure 4.2b illustrates this graph shell model. In the first iteration, there are no preceding layers, hence, the first expert processes the information in the same way as in the first model. In succeeding iterations, the aggregated representations are first sent to neighboring nodes, which in turn process the incoming representations. These aggregated values define the input for the current iteration. Full-colored shells illustrate aggregated values from previous iterations.

• **SEA-K-HOP.** For this model, the aggregate function defined in Equation 4.5 is relaxed. Given a graph $\mathcal{G}$, we also consider k-hop linkages in the graph connecting a node u with all entities having a maximum distance of $d_{\mathcal{G}}(u,v) = k$. The relaxation of Equation 4.5 is formalized as:

$$\hat{h}_u^{l+1} = O_h^l \Big\|_{i=1}^{H} \Big(\sum_{v \in N_k(u)} \underbrace{w_{uv}^{i,l}}_{\text{attention scores Eq. 4.7}} V^{i,l} \underbrace{h_v^l}_{\substack{\text{embds. of nodes} \\ \text{in k-hop dist.}}} \Big), \tag{4.11}$$

where $N_k(u)$ denotes the k-hop neighborhood set. This approach allows for processing each $N_1(u), \ldots, N_k(u)$ by own submodules, i.e., for each k-hop neighbors, respective feedforward networks are employed to compute Q, K, V in GTLs. Notably, Equation 4.11 can be interpreted as a generalization of the vanilla architecture, which is given by setting $k = 1$. Figure 4.2c shows the k-hop graph shell model with $k = 2$.

4.2.4.2 SEA: Routing Mechanism

By endowing the new models with experts referring to various graph shells, we gain variable expressiveness for short- and long-term dependencies. Originally introduced for language modeling and machine translation, Shazeer et al. [190] proposed a Mixture-of-Experts (MoE) layer. A routing module decides to which expert the attention is steered. This thesis uses a *single expert* strategy [57].

The general idea relies on a routing mechanism for a node u's representation to determine the best expert from a set $\{E_i(u)\}_{i=1}^N$ of N experts processing graph shells as described in the previous Section 4.2.4.1. The router module consists of a linear transformation whose output is

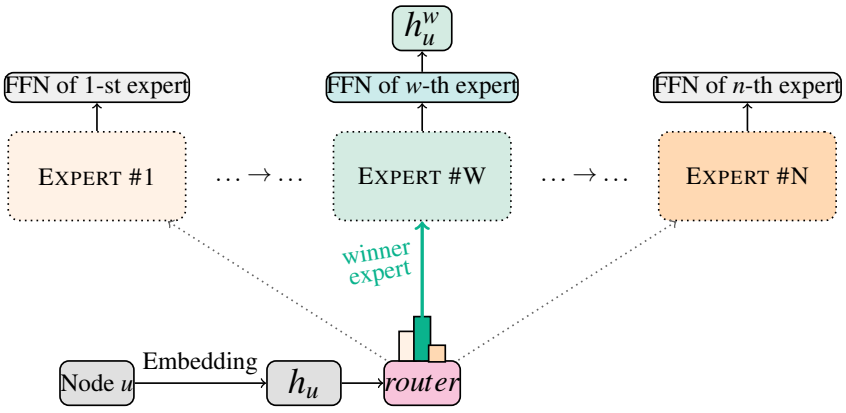


Figure 4.3: *Routing mechanism to N experts*

normalized via softmaxing. The probability of choosing the i-th expert for node u is defined as:

$$p_i(u) = \frac{\exp(r(u)_i)}{\sum_j^N \exp(r(u)_j)}, \quad r(u) = h_u^T W_r + b_r, \tag{4.12}$$

where $r(\cdot)$ denotes the routing operation with $W_r \in \mathbb{R}^{d \times N}$ being the routing's learnable weight matrix, and b_r denotes a bias term. The idea is to select the winner expert $E_w(\cdot)$ that is the most representative for a node's representation, i.e., where $w = \underset{i=1,\ldots,N}{\text{argmax}}\ p_i(u)$ [21]. A node's representation calculated by taking the winner's graph shells into account is then used as input for the expert's individual linear transformation:

$$h_u^w = E_w(u)^T W_w + b_w, \tag{4.13}$$

where $W_w \in \mathbb{R}^{d \times d}$ denotes the weight matrix of expert $E_w(\cdot)$, b_w denotes the bias term. The node's representation according to expert $E_w(\cdot)$, is denoted by h_u^w. Figure 4.3 shows how the routing is integrated into the novel architecture.

4.2.4.3 Shells vs. Over-smoothing

Over-smoothing in GNNs is a well-known issue [226] and exacerbates the problem when we build deeper graph neural net models. Applying the same number of iterations for each node inhibits the simultaneous expressiveness of short- and long-term dependencies. The novel workflow gains expressiveness by routing each node representation towards dedicated experts processing only nodes in their k-localized receptive field.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph. Following the proof scheme of [153], let $A =$

[21] In DL libraries, the argmax$(\cdot)$ operation implicitly calls max$(\dot{)}$ forwarding the maximum of the input. Hence, it is differentiable w.r.t. to the values yielded by the max op., not to the indices

$(\mathbb{1}_{(i,j)\in\mathscr{E}})_{i,j\in[N]:=\{1,\dots,N\}} \in \mathbb{R}^{N\times N}$ be the adjacency matrix and $D := \operatorname{diag}(\deg(i)_{i\in[N]}) \in \mathbb{R}^{N\times N}$ be the degree matrix of $\mathscr{G}$ where $\deg(i) := |\{j \in V \mid (i,j) \in \mathscr{E}\}|$ is the degree of node i. Let $\tilde{A} := A + I_N$, $\tilde{D} := D + I_N$ be the adjacent and the degree matrix of graph $\mathscr{G}$ augmented with self-loops, where I_N denotes the identity matrix of size N. The augmented normalized Laplacian of $\mathscr{G}$ is defined by $\tilde{\Delta} := I_N - \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$ and set $P := I_N - \tilde{\Delta}$. Let $L, C \in \mathbb{N}_+$ be the layer and channel sizes, respectively. W.l.o.g, for weights $W_l \in \mathbb{R}^{C\times C}(l \in [L] := \{1,\dots,L\})$, we define a GCN associated with $\mathscr{G}$ by $f = f_L \circ \dots \circ f_1$ where $f_l : \mathbb{R}^{N\times C} \to \mathbb{R}^{N\times C}$ is defined by $f_l(X) = \sigma(PXW_l)$, where $\sigma(\cdot)$ denotes the ReLU activation function. For $M \leq N$, let U be a M-dimensional subspace of $\mathbb{R}^N$. Furthermore, let $\mathscr{M}$ be a subspace of $\mathbb{R}^{N\times C}$ by $\mathscr{M} = U \otimes \mathbb{R}^C = \{\sum_{m=1}^{M} e_m \otimes w_m \mid w_m \in \mathbb{R}^C\}$, where $(e_m)_{m\in[M]}$ is the orthonormal basis of U. For an input $X \in \mathbb{R}^{N\times C}$, the distance between X and $\mathscr{M}$ is denoted by $d_{\mathscr{M}} = \inf\{\|X - Y\|_F \mid Y \in \mathscr{M}\}$.

Considering $\mathscr{G}$ as M connected components, i.e., $V = V_1 \cup \dots \cup V_m$, where an indicator vector of the m-th connected component is denoted by $u_m = (\mathbb{1}_{\{n\in V_m\}})_{n\in[N]} \in R^N$. The authors of [153] investigated the asymptotic behavior of the output X^L of the GCN when $L \to \infty$:

Proposition 1. *Let $\lambda_1 \leq \dots \leq \lambda_N$ be the eigenvalue of P sorted in ascending order. Then, we have $-1 < \lambda_1, \lambda_{N-M} < 1$, and $\lambda_{N-M+1} = \dots = \lambda_N = 1$. In particular, we have $\lambda = \max_{n=1,\dots N-M}|\lambda_n| < 1$. Further, $e_m = \tilde{D}^{\frac{1}{2}}u_m$ for $m \in [M]$ are the basis of the eigenspace associated with the eigenvalue* 1.

Let $s = \sup_{l\in\mathbb{N}_+} s_l$ with s_l denoting the maximum singular value of W_l, the major theorem and their implications for GCNs is stated as follows:

Theorem 1. *For any initial value $X^{(0)}$, the output of l-th layer $X^{(l)}$ satisfies $d_{\mathscr{M}}(X^{(l)}) \leq (s\lambda)^l d_{\mathscr{M}}(X^{(0)})$. In particular, $d_{\mathscr{M}}(X^{(l)})$ exponentially converges to* 0 *when $s\lambda < 1$.*

Proofs of Proposition 1 and Theorem 1 are formulated in [153].

Intuitively, the representations X align subsequently with the subspace $\mathscr{M}$, where the distance between both converges to zero. Therefore, it can also be interpreted as information loss of graph neural nets in the limit of infinite layers.

The theoretical justification for the routing mechanism applied in the novel SEA models comes to light when exploiting the monotonous behavior of the exponential decay where the initial distance $d_{\mathscr{M}}(X^{(0)})$ is treated as a constant value. The architecture includes the experts in a cascading manner, where the routing mechanism allows to point to each of the $(d_{\mathscr{M}}(f_l(X)))_{l=1,\dots,L}$, separately. From Theorem 1, we get:

$$\begin{aligned} d_{\mathscr{M}}(X^{(L)}) &\leq (s\lambda)^L d_{\mathscr{M}}(X^{(0)}) \leq (s\lambda)^{L-1} d_{\mathscr{M}}(X^{(0)}) \\ &\leq \dots \leq (s\lambda)^1 d_{\mathscr{M}}(X^{(0)}), \end{aligned}$$

Domain	Dataset	#Graphs	Task
Chemistry	ZINC	12K	Graph Regression
	OGBG-MOLHIV	41K	Graph Classification
Mathematical Modeling	PATTERN	14K	Node Classification

Table 4.1: *Summary dataset statistics*

where each inequality is supported by the output of the l-th expert, separately:

$$\begin{aligned}
&L\text{-th expert:} && d_{\mathscr{M}}(X^{(L)}) && \leq (s\lambda)^{L} d_{\mathscr{M}}(X^{(0)}) \\
&L\text{-1-th expert:} && d_{\mathscr{M}}(X^{(L-1)}) && \leq (s\lambda)^{L-1} d_{\mathscr{M}}(X^{(0)}) \\
&\dots && \dots && \leq \dots \\
&\text{1-st expert:} && d_{\mathscr{M}}(X^{(1)}) && \leq (s\lambda)^{1} d_{\mathscr{M}}(X^{(0)})
\end{aligned}$$

Hence, the architecture does not suffer from over-smoothing the same way as standard GNNs, as each captures a different distance $d_{\mathscr{M}}$ compared to using a GNN where a pre-defined number of layer updates is applied for all nodes equally and potentially leading to an over-smoothed representation.

4.2.5 Evaluation

4.2.5.1 Experimental Setting

Datasets.
ZINC [98] is one of the most popular real-world molecular dataset consisting of 250K graphs. A subset consisting of 10K train, 1K validation, and 1K test graphs is used in the literature as benchmark [50].

The thesis also evaluates the new models on **ogbg-molhiv** [91]. Each graph within the dataset represents a molecule, where nodes are atoms and edges are chemical bonds.

A benchmark dataset generated by the *Stochastic Block Model* (SBM) [1] is **PATTERN**. The graphs within this dataset do not have explicit edge features.
The benchmark datasets are summarized in Table 4.1.

Implementation Details.
The implementation for this thesis uses PyTorch, Deep Graph Library (DGL) [215], and OGB [91]. The models are trained on an NVIDIA GeForce RTX 2080 Ti. [22]

[22] Code: https://github.com/christianmaxmike/SEA-GNN

	ZINC	
Model	**#params.**	**MAE**
GCN [111]	505K	0.367
GIN [226]	509K	0.526
GAT [207]	531K	0.384
SAN [117]	508K	**0.139**
Graphormer-SLIM [228]	489K	**0.122**
Vanilla GTL	83K	0.227
SEA-GTL	347K	0.212
SEA-AGGREGATED	112K	0.215
SEA-2-HOP	**430K**	**0.159**
SEA-2-HOP-AUG	709K	0.189

(a) ZINC [98]

	OGBG-MOLHIV	
Model	**#params.**	**%ROC-AUC**
GCN-GRAPHNORM [111]	526K	76.06
GIN-VN [226]	3.3M	77.80
DGN [14]	114K	79.05
Graphormer-FLAG [228]	47.0M	**80.51**
Vanilla GTL	386K	78.06
SEA-GTL	347K	79.53
SEA-AGGREGATED	**133K**	**80.18**
SEA-2-HOP	511K	**80.01**
SEA-2-HOP-AUG	594K	79.08

(b) ogbg-molhiv [91]

	PATTERN	
Model	**#params.**	**% ACC**
GCN [111]	500K	71.892
GIN [226]	100K	85.590
GAT [207]	526K	78.271
GraphSage [79]	101K	50.516
SAN [117]	454K	**86.581**
Vanilla GTL	82K	84.691
SEA-GTL	132K	85.006
SEA-AGGREGATED	69K	57.557
SEA-2-HOP	**48K**	**86.768**
SEA-2-HOP-AUG	152K	**86.673**

(c) PATTERN [1]

Table 4.2: *Comparison to state-of-the-art; results are partially taken from [117, 50]; color coding (gold/silver/bronze)*

Model Configuration.
The implementation uses the Adam optimizer [110] with an initial learning rate $\in \{1\text{e-}3, 1\text{e-}4\}$. It applies the same learning rate decay strategy for all models that half the learning rate if the validation loss does not improve over a fixed number of 5 epochs. It tunes the pairing $(\#\text{heads}, \text{hidden dimension}) \in \{(4,32),(8,56),(8,64))\}$ and uses READOUT $\in \{sum\}$ as function for inference on the whole graph information. *Batch Normalization* and *Layer Normalization* are disabled, whereas residual connections are activated per default in GTLs. For dropout, the value is tuned to be $\in \{0, 0.01, 0.05, 0.07, 0.1\}$ and a weight decay $\in \{5\text{e-}5, 5\text{e-}7\}$. For the number of graph shells, i.e., number of experts being used, the thesis reports values $\in \{4,6,8,10,12\}$. As aggregation function the implementation uses AGG $\in \{mean\}$ for Equation 4.10. As laplacian encoding, the 8 smallest eigenvectors are used.

4.2.5.2 Prediction Tasks

In the following series of experiments, the thesis investigates the performance of the *Graph Shell Attention* mechanism on graph-level prediction tasks for the datasets ogbg-molhiv [91] and ZINC [98], and a node-level classification task on PATTERN [1]. The evaluation uses commonly used metrics for the prediction tasks as they are used in [50], i.e., mean absolute error (MAE) for ZINC, the ROC-AUC score on ogbg-molhiv, and the accuracy on PATTERN.

Competitors. The thesis evaluates the new architectures against state-of-the-art GNN models achieving competitive results. The report subsumes the vanilla GCN [111], GAT [207] that includes additional attention heuristics, or more recent GNN architectures building on top of Transformer-enhanced models like SAN [117] and Graphormer [228]. Moreover, it includes GIN [226] that is more discriminative towards graph structures compared to GCN [111], GraphSage [79], and DGN [14] being more discriminative than standard GNNs w.r.t. the Weisfeiler-Lehman 1-WL test.

Results. Tables 4.2a, 4.2b, and 4.2c summarize the performances of the SEA models compared to baselines on ZINC, ogbg-molhiv, and PATTERN. *Vanilla GTL* shows the results of the implementation of the GNN model including Graph Transformer Layers [49]. *SEA*-2-HOP includes the 2-hop connection within the input graph, whereas *SEA*-2-HOP-AUG process the input data the same way as the 2-HOP heuristic, but uses additional feedforward networks for computing Q, K, V values for the 2-hop neighbors.

For PATTERN, we observe the best result using the *SEA*-2-HOP model, beating all other competitors. On the other hand, distributing an aggregated value to neighboring nodes according to *SEA*-AGGREGATED yields a too coarse view for graphs following the SBM and loses local graph structure.

In the sense of *Green AI* [186] that focuses on reducing the computational cost to encourage a reduction in resources spent, the novel architecture reaches state-of-the-art performance on ogbg-molhiv while drastically reducing the number of parameters being trained. Comparing *SEA*-AGGREGATED to the best result reported for *Graphormer* [228], the new model economizes on **99**.**71**% of the number of parameters while still reaching competitive results.

The results on ZINC enforces the argument of using individual experts compared to vanilla GTLs, where the best result is reported for *SEA*-2-HOP.

4.2.5.3 Number of Shells

Next, the thesis examines the performance w.r.t. the number of experts. Notably, increasing the number of experts correlates with the number of *Graph Shells* which are taken into account. Table 4.3 summarizes the results where all other hyperparameters are frozen, and we only have a variable size in the number of experts. Each model is trained for 500 epochs and the best-observed metrics on the test datasets are reported. The implementation applies an early stopping heuristic, where the learning procedure is stopped if no improvements w.r.t. the evaluation metrics were observed or if the learning rate scheduler reaches a minimal value which is set to 10^{-6}. Each evaluation on the test data is conducted after 5 epochs, and the early stopping is effective after 10 consecutive evaluations on the test data with no improvements. First, note that increasing the number of experts also increases the model's parameters linearly. This is due to additional routings and linear layer being defined for each expert separately. Secondly, the thesis reports also the average running time in seconds $[s]$ on the training data for each epoch. By construction, the running time correlates with the number of parameters that have to be trained. The number of parameters differs from one dataset to another with the same settings due to a different number of nodes and edges within the datasets and slightly differs if biases are used or not. Note that we observe better results of *SEA*-AGGREGATED by decreasing the embedding size from 64 to 32, which also applies for the PATTERN dataset in general. The increase of parameters of the augmented 2-hop architecture *SEA*-2-HOP-AUG is due to the additional feedforward layers

Model	#experts	ZINC #params	ZINC MAE	ZINC time/ epoch	OGBG-MOLHIV #params.	OGBG-MOLHIV %ROC-AUC	OGBG-MOLHIV time/ epoch	PATTERN #params.	PATTERN % ACC	PATTERN time/ epoch
SEA-GTL	4	183K	0.385	13.60	182K	79.24	49.21	48K	78.975	58.14
	6	266K	0.368	20.93	263K	78.24	68.67	69K	82.117	82.46
	8	349K	**0.212**	26.24	345K	**79.53**	84.35	90K	82.983	108.41
	10	433K	0.264	31.63	428K	79.35	107.11	111K	84.041	133.73
	12	516K	0.249	38.26	511K	79.18	122.99	132K	**85.006**	168.47
SEA-AGGREGATED	4	49K	0.257	31.24	48K	77.87	60.98	48K	57.490	99.10
	6	70K	0.308	44.61	69K	79.21	86.26	69K	**57.557**	106.79
	8	91K	0.249	57.89	90K	77.19	86.93	90K	54.385	131.57
	10	112K	**0.215**	73.49	111K	77.48	102.40	111K	57.221	173.74
	12	133K	0.225	87.08	132K	**80.18**	124.08	132K	57.270	206.73
SEA-2-HOP	4	182K	0.309	14.28	180K	76.30	43.51	48K	**86.768**	94.04
	6	265K	0.213	20.13	263K	77.27	59.82	69K	86.706	138.10
	8	347K	0.185	24.91	345K	76.61	79.56	90K	86.707	178.64
	10	430K	**0.159**	32.68	428K	78.38	95.69	111K	86.680	232.91
	12	513K	0.188	38.73	511K	**80.01**	112.93	132K	86.699	269.71
SEA-2-HOP-AUG	4	248K	0.444	16.86	248K	77.21	48.65	65K	84.889	124.96
	6	363K	0.350	24.84	363K	75.19	70.05	94K	85.141	203.38
	8	478K	0.285	31.48	476K	76.55	90.78	123K	86.660	270.85
	10	594K	0.205	39.25	594K	**79.08**	109.91	152K	**86.673**	363.58
	12	709K	**0.189**	46.51	707K	77.52	133.48	181K	86.614	421.46

Table 4.3: *Influence of the number of experts applied on various SEA models; best configurations are highlighted in green*

being used for the k-hop neighbors to compute the inputs Q, K, V in the graph transformer layer. Notably, we also observe that similar settings apply for datasets where the structure is an important feature of the graph, like in molecules (ZINC + ogbg-molhiv). In contrast to that is the behavior on graphs following the stochastic block model (PATTERN). On the latter one, the best performance could be observed by including k-hop information, whereas an aggregation yields too simplified features to be competitive. For the real-world molecules (ZINC + ogbg-molhiv) datasets, the evaluation revealed that more experts boost the performance for the various SEA extensions.

Model	#exp.	k	ZINC #prms	ZINC MAE	OGBG-MOLHIV #prms	OGBG-MOLHIV %ROC-AUC	PATTERN #prms	PATTERN %ACC
SEA-K-HOP	6	2	265K	0.213	263K	**77.27**	69K	**86.768**
		3	266K	**0.191**	263K	76.15	69K	86.728
		4	266K	0.316	263K	73.48	69K	86.727
	10	2	430K	**0.159**	428K	**78.38**	111K	86.680
		3	433K	0.171	428K	74.67	111K	**86.765**
		4	433K	0.239	428K	73.72	111K	86.725

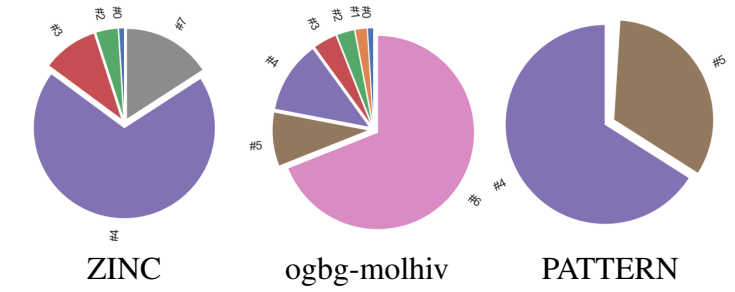
Table 4.4: *Influence of parameter k for the* SEA-K-HOP *model; best configuration for each model is highlighted in green*

4.2.5.4 Stretching Locality in SEA-K-HOP

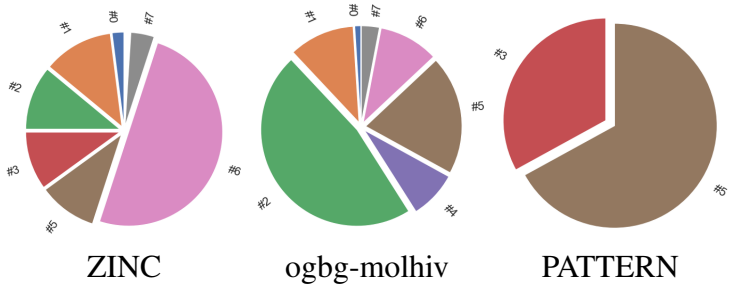
In the following, the thesis investigates the influence of the parameter k for the *SEA*-K-HOP model. Generally, by increasing the parameter k, the model diverges to the full model being also examined for the *SAN* architecture explained in [117]. In short, the full setting takes edges into account that is given by the input data and also sends information over non-existent edges, i.e., the argumentation is on a full graph setting. In the proposed architecture, the transition is smoothed from edges being given in the input data to the full setting that naturally arises when k, the number of hops, is set to a sufficiently high number. Table 4.4 summarizes the results for the non-augmented model, i.e., no extra linear layers are used for each k-hop neighborhood. The number of parameters stays the same by increasing k.

4.2.5.5 Distribution of Experts

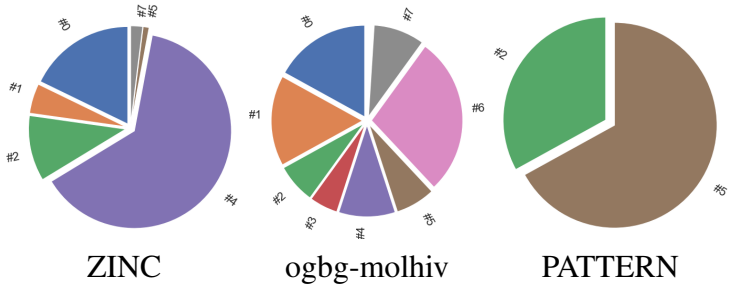
Lastly, the thesis evaluates the distributions of the experts being chosen to compute the nodes' representations. The number of experts is set to 8. Figure 4.4 summarizes the relative frequencies of the experts being chosen on the datasets ZINC, ogbg-molhiv, and PATTERN. Generally, the performance of the shell attention heuristic degenerates whenever *expert collapsing* can be observed. In the extreme case, just one expert expresses the mass of all nodes, and the capability to distribute nodes' representations over several experts is not leveraged. To overcome *expert collapsing*, a heuristic can be used where in the early stages of the learning procedure, an additional epsilon parameter ε introduces randomness. Like a decaying greedy policy in Reinforcement Learning (RL), the framework chooses a random expert with probability ε and choose the expert with the highest probability according to the routing layer with a probability of $1-\varepsilon$. The epsilon value slowly decays over time. This ensures that all experts' expressiveness is being explored to find the best matching one w.r.t. to a node u and prevents getting stuck in a local optimum. The figure shows the distribution of experts that are relevant for the computation of the nodes' representations. For illustrative purposes, values below 1% are omitted. Generally, nodes are more widely distributed over all experts in the molecular datasets - ZINC and ogbg-molhiv - for all models compared to PATTERN following a stochastic block model. Therefore, various experts are capable of capturing individual topological characteristics of molecules better than vanilla graph neural networks for which over-smoothing might potentially occur. We also observe that the mass is distributed to only a subset of the available experts for the PATTERN dataset. Hence, the specific number of iterations is more expressive for nodes within graph structures following SBM.



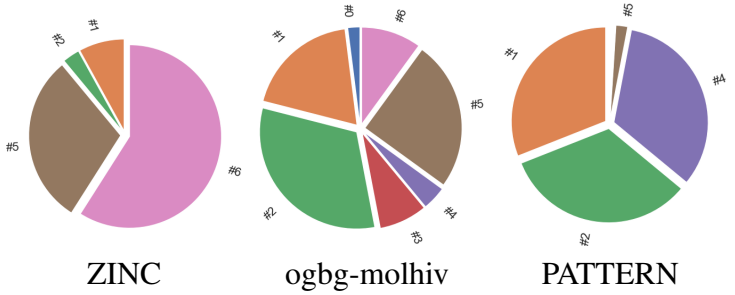
(a) SEA-GNN



(b) SEA-AGGREGATED



(c) SEA-2-HOP



(d) SEA-2-HOP-AUG

Figure 4.4: *Distribution of 8 experts for models* SEA-GNN, SEA-AGGREGATED, SEA-2-HOP, *and* SEA-2-HOP-AUG *for datasets ZINC, ogbg-molhiv and PATTERN. Relative frequencies are shown for values* $\geq 1\%$*. Numbers attached to the slices refer to the respective experts.*

4.3 Summary

The thesis introduced the theoretical foundation for integrating an expert heuristic within transformer-based graph neural networks. This opens a fruitful direction for future works that go beyond successive message-passing to develop even more powerful architectures in graph learning.

The thesis provides an engineered solution that allows selecting the most representative experts for nodes in the input graph. For that, the novel model exploits the idea of a routing layer steering the nodes' representations towards the individual expressiveness of dedicated experts. As experts process different subgraphs starting from a central node, the terminology of *Graph **S**hell **A**ttention* (SEA) is introduced, where experts solely process nodes that are in their respective receptive field. Therefore, additional expressiveness is gained by capturing varying short- and long-term dependencies expressed by individual experts. In a thorough experimental study, the thesis shows on real-world benchmark datasets that the gained expressiveness yields competitive performance compared to state-of-the-art results while being more economically. Additionally, it reports experiments that stress the number of graph shells that are taken into account.

A potent research direction is to work on more novel implementations and applications enhanced with the graph shell attention mechanism, e.g., where different hyperparameters are used for different graph shells.

5

Probabilistic Graphs

The most important questions of life are indeed, for the most part, really only problems of probability.

Pierre-Simon Laplace
1749 - 1827

Attribution

This Chapter uses material from the following publications:

- Christian Frey, Andreas Züfle, Tobias Emrich, and Matthias Renz. Efficient information flow maximization in probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30:880–894, 2018. [61]
- Christian Frey, Andreas Züfle, Tobias Emrich, and Renz. Efficient information flow maximization in probabilistic graphs (extended abstract). In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1801–1802. IEEE Computer Society, 2018. doi: 10.1109/ICDE.2018.00258. URL `https://doi.org/10.1109/ICDE.2018.00258` [60]
- Christian M. M. Frey, Andreas Züfle, Tobias Emrich, Matthias Renz, and Regine Meunier. Efficient data propagation in a computer network. `https://pubchem.ncbi.nlm.nih.gov/patent/US-2020394249-A1`, 2016. Accessed: 2022-9-24 [63]

See § 1.4 for an overview of incorporated publications and the author's attribution.

Highlights

- Theoretical complexity study of the flow maximization problem in probabilistic graphs;
- Efficient estimation of the expected information flow based on network graph decomposition and Monte-Carlo sampling;
- The proposed *F-tree* structure enables an efficient organization of independent graph components and (local) intermediate results for an efficient expected flow computation;
- Algorithms for an iterative selection of edges to be activated to maximize the expected information flow;
- Thorough experimental evaluation of proposed algorithms;

5.1 Data Analytics on Probabilistic Graphs

In the analysis of graphs it becomes an integral part to analyze the structure of the data using various algorithms and apply procedures to shape the data such that we can extract knowledge from it. Generally, Data Analytics is essential for extracting hidden patterns and gaining knowledge from data. Choosing the appropriate technical tools and for solving analytical questions and problems is the key differentiator in the economic competition. In recent years, the advancements in Artificial Intelligence (AI) and its Machine Learning and Deep Learning models is the predominant development in solving special downstream tasks such as classification, clustering, regression, recognition, pattern detection, association, and so forth. However, it starts with an appropriate organization of the input data to process diverse tasks more efficiently. In the following, the thesis introduces a novel data structure in the scope of probabilistic graphs, i.e., graphs, where probability values are imposed on the connections of the graph. First, the chapter provides an overview of different scenarios where probability values play an essential role in real-world scenarios as the reliability of a graph system is often assumed to be given.

Fields of Application of Probabilistic Networks:

Social Networks.
Research topics in the field of modeling probabilistic graphs revealed innovative access in creating and understanding the emergence and structure of social networks. In recent years a growing number of scientists steered their attention on variety settings of social networks. Those networks can be recognized in many disciplines of social, political and economic life (cf. [158]). Nevertheless, the calculations of interactions between the involved parties can all be accomplished in the same manner. The underlying problem just expresses itself in different ways in diverse realms, i.e., it can be expressed as the dissemination of knowledge/information, as the exertion of power/influence or as the interplay between companies and customers in the area of viral marketing. With the opportunities of measuring and monitoring interactions in a social network over a variable period of time, it also found its way in the sphere of social science.
Figure 5.1 shows the concept of viral marketing expressed within a social network. The company reaches potential customers through their advertising. One strategy could be to identify nodes, e.g., other companies or influential persons, in order to acquire as much new customers as possible. Therefore, it is needed to identify important links in the network. Further concepts including the cooperation between various companies could be expressed with probabilities on the links. One could formulate a statistical view of potential collaborations in the future. The understanding of reliable dynamic developments in economic processes can be supported by a sophisticated analysis of social networks.

Genetic, Genomics and Postgenomics.
A fundamental problem in biology is to understand the structure and functional organization of genomes. Likewise to other research fields, new technologies allow to collect more data. As a result, an increase in demand for large-scale statistical methods came up. These can be applied on so-called '*omics*' databases ('omics'= a neologism denoting data of genomics, proteomics or metabolomics databases) and other types of biological data (cf. [192]). The processes in chemical biology is expressed in dependency or independency in the collected data. The analysis can be conducted in a discrete point of time or in a continuous context. Links can be made between millions of correlated observations from a single individual at different scales, i.e., examining observations on a molecular level up to an atomic one. By studying living systems, analytics can answer questions referring prospective expectations or causal issues. With the data being represented in a graphical model, the understanding of biological mechanisms, predicting outcomes and deciphering causal relationships can be facilitated. The possibility to express dependencies within the data being one of the most defining feature of cellular and other biological data, is of utmost importance.

Transportation Networks.
As web services such as *"Google Maps"*, "Yahoo! maps" or *"Open Street Map"* came up with answering queries like *"Finding the shortest path in travel time from point A to point B"*, new emerging techniques for traffic monitoring were included in this analysis. By collecting data of roadside sensors or cell phone signals, more traffic data about road networks became available (cf. [93]).

Data aggregated from roadside sensors include information about the traffic volume, vehicle speeds etc. It is also possible to classify vehicles coinciding with certain criteria. The main drawback on this stage is that the collected data cannot guarantee its total correction, i.e., analytics have to take a note of the limitations of necessary equipment and the delay or even loss of data while transferring it to a root system. Analyzing the collected data, one should also take into account the differences of the vehicles' speed. Therefore, the travel time along each road segment derived from the collected data of sensors are inherently uncertain and are based on statistical calculations. Modeling a road network (see Figure 5.2) can be quite efficient using a graph system. It is quite common to project the travel time onto edge's weights in a road network. To express its uncertainty, one can use random variables. A road network including statistical information about the traffic is often referred to as an *uncertain traffic network*.

Ad-hoc Networks.
An Ad-hoc network connects temporarily diverse devices communicating with each other for a certain purpose, i.e., sharing documents, playing multi-player games or providing Internet access. The communication is accomplished by using wireless transceivers without the need for a fixed infrastructure. Compared with more traditional networks like a wireless LAN, in which

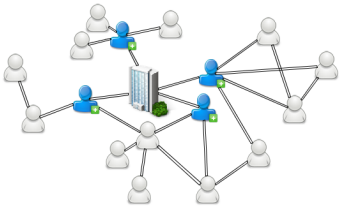


Figure 5.1: *Example of viral marketing.*

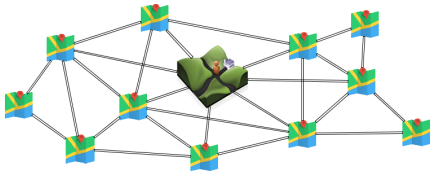


Figure 5.2: *Example of a road network*

the included devices need to communicate through base stations, an ad-hoc network is of dynamic nature. In such a network the nodes are devices, i.e., laptops, smart phones, tablets etc., and communicating components are linked with each other (cf. [179]). Partially established in public places (e.g., airport, shopping malls, ...), mobile devices can make Internet in combination with base stations ubiquitous accessible (see Figure 5.3). In order to keep up a network, links between devices have to be managed with the result that the communication within the included components is never interrupted. Therefore, the system has to fulfill the task of maintaining the information flow between the nodes. For now, it is also subject of research to develop stable ***Vehicular Ad Hoc Network*** (VANet for short). In this context, nodes are vehicles or so-called **R**oad-**S**ide-**U**nits (RSU). A VANet has a high dynamic topology. Again, the task is to establish a mobile Internet providing the ability that the vehicles could communicate with each other simultaneously. This might be of interest for safety purposes, e.g., rescue service, fire departments, police cars, etc.

WSN - Wireless Sensor Networks.
A special type of ad-hoc networks are ***Wireless Sensor Networks*** (WSN). The nodes of such a network - also referred to as a communication graph - are small sensors (see Figure 5.4). A WSN is typically composed of components providing same features. Incorporated functionalities allow it to record, for example, thermal conditions, pressure measurements or register acoustic signals. The sensors are also equipped with small processors being able to transmit or receive information wireless at short-range. In difference to an ad-hoc network the sensors are in most applications stationary. Also the number of nodes in a WSN is quite large giving a global view of a monitored region. The sensors are exchanging information which is made accessible to an user through one or more so-called gateway node(s). These gateways are designated sink nodes (source, base station) of a sensor network. The reliability of a sensor network depends on the set of wireless links connecting the nodes and, therefore, enabling the communication within the integrated sensors. A link between two components can depend on *i)* the relative distance between them, *ii)* the transmit power used to send the data, or *iii)* the surrounding environment (cf. [179]). Considering a WSN observing the thermal conditions of a forested area. In order to detect outbreaks of fire, the sensors are communicating anomalous values to adjacent nodes. By

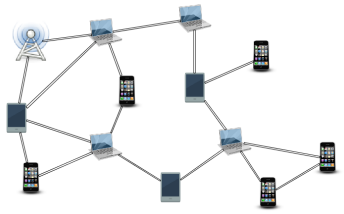


Figure 5.3: *Example of viral marketing.*

Figure 5.4: *Example of a road network*

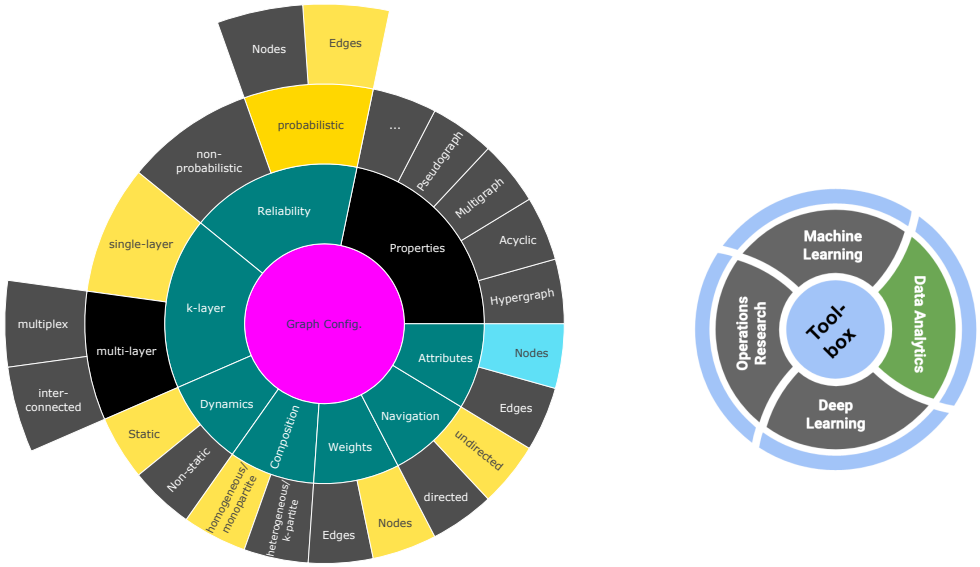
analyzing the recorded values of different sensors, the fire can be located geographically. This evaluated information is transmitted to a gateway, where the ranger is informed about the forest fire. One question arising in this scenario is *how to ensure that the information is transmitted to the gateway? Which sensors need to communicate with each other maximizing the probability that the fire will be detected? Which links need to be established to increase the system's reliability?* Answers to these questions can be given by analyzing the graph's topology and direct one's attention to the optimization of information flow in probabilistic graphs.

5.2 Flow Tree for Probabilistic Graphs

Reliable propagation of information through large networks, e.g., communication networks, social networks or sensor networks is very important in many applications concerning marketing, social networks, and wireless sensor networks. However, social ties of friendship may be obsolete, and communication links may fail, inducing the notion of uncertainty in such networks. This thesis addresses the problem of optimizing information propagation in uncertain networks given a constrained budget of edges. It is shown that this problem requires to solve two NP-hard subproblems: the computation of expected information flow, and the optimal choice of edges. To compute the expected information flow to a source vertex, the *F-tree* is proposed as a specialized data structure, that identifies independent components of the graph for which the information flow can either be computed analytically and efficiently, or for which traditional Monte-Carlo sampling can be applied independently of the remaining network. For the problem of finding the optimal edges, the thesis proposes a series of heuristics that exploit properties of this data structure. The evaluation shows that these heuristics lead to high quality solutions, thus, yielding high information flow, while maintaining low running time.

Graph Configuration.
The configuration used for the novel data structure is in the realm of single-layer, static, and homogeneous graphs shown in Figure 5.5a. Additionally, probability values are attached to the undirected edges to express the reliability of the linkage between two nodes. The transmission of information units is expressed via weights being imposed on the nodes in the graph. In the eval-

uation, transportation networks are discussed where nodes within the graph have attribute values about the coordinates (highlighted in blue in the configurator). The proposed data structure to compute the information flow in a probabilistic setting is settled in the technical domain of *Data Analytics*.

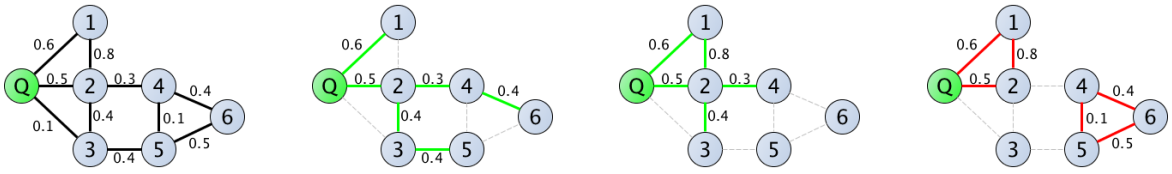


(a) *Graph Configurator* ***(b)*** *Toolbox*

Figure 5.5: *Graph Configuration and methodology's classification used to solve the efficient maximization information flow problem in probabilistic graphs by an Flow tree (F-tree)*

5.2.1 Motivation

Nowadays, social and communication networks have become ubiquitous in our daily life to receive and share information. Whenever we are navigating the *World Wide Web* (WWW), updating our social network profiles, or sending a text message on our cell-phone, we participate in an information network as a node. In such settings, network nodes exchange some sort of information: In social networks, users share their opinions and ideas, aiming to convince others. In *Wireless Sensor Networks* (WSN), nodes collect data and aim to ensure that this data is propagated through the network: Either to a destination, such as a server node, or simply to as many other nodes as possible. Abstractly speaking, in all of these networks, nodes aim at propagating their information, or their belief, throughout the network. The event of a successful propagation of information between nodes is subject to inherent uncertainty. In a wireless sensor, telecommunication or electrical network, a link can be unreliable and may fail with certain probability [72, 175]. In a social network, trust and influence issues may impact the likelihood



(a) original graph *(b) Dijkstra MST* *(c) Optimal five-edge flow* *(d) possible world g_1*

Figure 5.6: *Running example.*

of social interactions or the likelihood of convincing another of an individual's idea [77, 108, 2]. For example, consider professional social networks like *LinkedIn*. Such networks allow users to endorse each others' skills and abilities. Here, the probability of an edge may reflect the likelihood that one user is willing to endorse another user. The probabilistic graph model is commonly used to address such scenarios in a unified way (e.g., [133, 155, 162, 235, 229, 114, 236]). In this model, each edge is associated with an existential probability to quantify the likelihood that this edge exists in the graph. Traditionally, to maximize the likelihood of a successful communication between two nodes, information is propagated by flooding it through the network. Thus, every node that receives a bit of information will proceed to share this information with all its neighbors. Clearly, such a flooding approach is not applicable for large communication and social networks, as the communication between two network nodes incurs a cost: Sensor network nodes have limited computing capability, memory resources and power supply, but require battery power to send and receive messages, and are also limited by their bandwidth; individuals of a social network require time and sometimes even additional monetary resources to convince others of their ideas. For instance, a professional networking service may provide, for a fee, a service to directly ask a limited number of users to endorse another user Q. The challenge is to maximize the expected number of endorsements that Q will receive, while limiting the budget of users asked by the service provider. The first candidates to ask are Q's direct connections. In addition, if a user u has already endorsed Q, then u's connections can be asked if they trust u's judgment and want to make the same endorsement.

This thesis addresses the following problem: Given a probabilistic network graph $\mathcal{G}$ with edges that can be activated, i.e., enabled to transfer information, or stay inactive. The problem is to send/receive information from a single node Q in $\mathcal{G}$ to/from as many nodes in $\mathcal{G}$ as possible assuming a limited budget of edges that can be activated. To solve this problem, the main focus is on the selection of edges to be activated.

Example 1. *To illustrate the problem setting, consider the network depicted in Figure 5.6a. The task is to maximize the information flow to node Q from other nodes given a limited budget of edges. This example assumes equal weights of all nodes. Each edge of the network is labeled with the probability of a successful communication. A naive solution is to activate all edges. Assuming each node to have one unit of information, the expected information flow of this solution can be*

shown to be $\simeq 2.51$. *While maximizing the information flow, this solution incurs the maximum possible communication cost. A traditional trade-off between these single-objective solutions is using a probability maximizing* Dijkstra's *MST, as depicted in Figure 5.6b. The expected information flow in this setting can be shown to aggregate to* 1.59 *units, while requiring six edges to be activated. Yet, it can be shown that the solution depicted in Figure 5.6c dominates this solution: Only fives edges are used, thus further reducing the communication cost, while achieving a higher expected information flow of* $\simeq 2.02$ *units of information to Q.*

The aim of this thesis is to efficiently find a near-optimal sub-network, which maximizes the expected flow of information at a constrained budget of edges. In Example 1, the information flow for an example graph is computed. But in fact, this computation has been shown to be exponentially hard in the number of edges of the graph and, thus, impractical to be solved analytically. Furthermore, the optimal selection of edges to maximize the information flow is shown to be *NP*-hard. These two subproblems define the main computational challenges addressed in this thesis. To tackle these challenges, the remainder of this chapter is organized as follows. After a survey of related work in Section 5.2.2, the thesis recapitulates common definitions for stochastic networks and formally defines the problem setting in Section 5.2.3. After a more detailed technical overview in Section 5.2.4, the theoretical heart of this work is presented in Section 5.2.5. The thesis shows how to identify independent subgraphs, for which the information flow can be computed independently. This allows to divide the main problem into much smaller subproblems. To conquer these subproblems, the approach identifies cases for which the expected information flow can be computed analytically, and it proposes to employ Monte-Carlo sampling to approximate the information flow of the remaining cases. Section 5.2.5.3 is the algorithmic core of the chapter, showing how aforementioned independent components can be organized hierarchically in an *F-tree* which is inspired by the *block-cut tree* [199, 90, 222]. This structure allows to aggregate results of individual components efficiently, and the thesis shows how previous Monte-Carlo sampling results can be re-used as more edges are selected and activated. The experimental evaluation in Section 5.2.7 shows that the novel algorithms significantly outperform traditional solutions, in terms of combined communication cost and information flow, on synthetic and real stochastic networks.

5.2.2 Related Work

Reliability and Influence computation in probabilistic graphs (a.k.a. uncertain graphs) has recently attracted much attention in the data mining and database research communities. In the following, state-of-the-art publications are summarized and the new model is related in this context.

Subgraph Reliability. A related and fundamental problem in uncertain graph mining is the so-called subgraph reliability problem, which asks to estimate the probability that two given (sets of) nodes are reachable. This problem, well studied in the context of communication networks,

has seen a recent revival in the database community due to the need for scalable solutions for big networks. Specific problem formulations in this class ask to measure the probability that two specific nodes are connected (two-terminal reliability [3]), all nodes in the network are pairwise connected (all-terminal reliability [188]), or all nodes in a given subset are pairwise connected (k-terminal reliability [86, 80]). Extending these reliability queries, where source and sink node(s) are specified, the corresponding graph mining problem is to find, for a given probabilistic graph, the set of most reliable k-terminal subgraphs [106]. All these problem definitions have in common that the set of nodes to be reached is predefined, and that there is no degree of freedom in the number of activated edges - thus, all nodes are assumed to attempt to communicate to all their neighbors, which can be overly expensive in many applications.

Reliability Bounds. Several lower bounds on (two-terminal) reliability have been defined in the context of communication networks [22, 25, 68, 166]. Such bounds could be used in the place of the proposed sampling approach, to estimate the information gain obtained by adding a network edge to the current active set. However, for all these bounds, the computational complexity to obtain these bounds is at least quadratic in the number of network nodes, making these bounds unfeasible for large networks. Very simple but efficient bounds have been presented in [109], such as using the most-probable path between two nodes as a lower bound of their two-terminal reliability. However, the number of possible (non-circular) paths is exponentially large in the number of edges of a graph, such that in practice, even the most probable path will have a negligible probability, thus, yielding a useless upper bound. Thus, since none of these probability bounds are sufficiently effective and efficient for practical use, the proposed data structure uses a sampling approach for parts of the graph where no exact inference is possible.

Influential Nodes. Existing work motivated by applications in marketing provide methods to detect *influential* members within a social network. This can help to promote a new product. The task is to detect nodes, i.e., persons, where the chance that the product is recommended to a broad range of connected people is maximized. In [43], [169] a framework is provided which considers the interactions between the persons in a probabilistic model. As the problem of finding the most influential vertices is *NP-hard*, approximation algorithms are used in [108] outperforming basic heuristics based on degree centrality and distance centrality which are applied traditionally in social networks. This branch of research has in common that the task is to activate a constrained number of *nodes* to maximize the information flow, whereas the problem definition in this thesis constrains the number of activated *edges* for a single specified query/sink node.

Reliable Paths. In mobile ad hoc networks, the uncertainty of an edge can be interpreted as the connectivity between two nodes. Thus, an important problem in this field is to maximize the probability that two nodes are connected for a constrained budget of edges [72]. In this work, the main difference to the thesis' work is that the information flow to a single destination is maximized, rather than the information flow in general. The heuristics [72] cannot be applied directly to the problem stated in this thesis, since clearly, maximizing the flow to one node may detriment the flow to another node.

Bi-connected components. The *F-tree* that is proposed in this thesis is inspired by the *block-cut tree* [199, 90, 222]. The main difference is that the novel approach aims at finding cyclic subgraphs, where nodes are bi-connected. For subgraphs having a size of at least three vertices, this problem is equivalent to finding bi-connected subgraphs, which is solved in [199, 90, 222]. Thus, the proposed data structure treats bi-connected subgraphs of size less than three separately, grouping them together as mono-connected components. More importantly, this existing work does not show how to compute, estimate and propagate probabilistic information through the structure, which is the main contribution of this thesis.

5.2.3 Problem Definition

A probabilistic undirected graph is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W, P)$, where V is a set of vertices, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges, $W : \mathcal{V} \mapsto \mathbb{R}^+$ is a function that maps each vertex to a positive value representing the information weight of the corresponding vertex and $P : \mathcal{E} \mapsto (0,1]$ is a function that maps each edge to its corresponding probability of existing in $\mathcal{G}$. Hence, Definition 11 is extended by a weighting function. In the following, it is assumed that the existence of different edges are independent from one another. Let us note, that the proposed approach also applies to other models such as the conditional probability model [162], as long as a computational method for an unbiased drawing of samples of the probabilistic graph is available.

In a probabilistic graph $\mathcal{G}$, the existence of each edge is a random variable. Thus, the topology of $\mathcal{G}$ is a random variable, too. The sample space of this random variable is the set of all *possible graphs*. A *possible graph* $g = (\mathcal{V}_g, \mathcal{E}_g)$ of a probabilistic graph $\mathcal{G}$ is a deterministic graph which is a possible outcome of the random variables representing the edges of $\mathcal{G}$. The graph g contains a subset of edges of $\mathcal{G}$, i.e. $\mathcal{E}_g \subseteq \mathcal{E}$. The total number of possible graphs is $2^{|E_{<1}|}$, where $|\mathcal{E}_{<1}|$ represents the number of edges $e \in \mathcal{E}$ having $P(e) < 1$, because for each such edge, we have two cases as to whether or not that edge is present in the graph. Let $\mathcal{W}$ denote the set of all possible graphs. The probability of sampling the graph g from the random variables representing the probabilistic graph $\mathcal{G}$ is given by the following sampling or realization probability $Pr(g)$:

$$Pr(g) = \prod_{e \in \mathcal{E}_g} P(e) \cdot \prod_{e \in \mathcal{E} \setminus \mathcal{E}_g} (1 - P(e)). \tag{5.1}$$

Figure 5.6a shows an example of a probabilistic graph $\mathcal{G}$ and one of its possible realization g_1 in 5.6d. This probabilistic graph has $2^{10} = 1024$ possible worlds. Using Equation 5.1, the probability of world g_1 is given by:

$$\begin{aligned} Pr(g_1) =& 0.6 \cdot 0.5 \cdot 0.8 \cdot 0.4 \cdot 0.4 \cdot 0.5 \cdot (1-0.1) \cdot \\ & \cdot (1-0.3) \cdot (1-0.4) \cdot (1-0.1) = 0.00653184 \end{aligned}$$

Definition 18 (Path)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W, P)$ be a probabilistic graph and let $v_0, v_n \in \mathcal{V}$ be two*

nodes such that $v_0 \neq v_n$. An (acyclic) ***path*** *$path(v_0, v_n) = (v_0, v_1, v_2, \ldots, v_n)$ is a sequence of vertices, such that $(\forall v_i \in path(v_0, v_n))(v_i \in V)$ and $(\forall v_i \in path(v_0, v_{n-1}))((v_i, v_{i+1}) \in \mathscr{E})$.*

Notably, in contrast to Definition 3, a *path* in Definition 18 is based on the sequence of vertices rather than on the sequence of edges being traversed.

Definition 19 (Reachability)**.** *The network reachability problem as defined in [101, 32] computes the likelihood of the binomial random variable $\updownarrow(i, j, \mathscr{G})$ of two nodes $i, j \in \mathscr{V}$ being connected in $\mathscr{G}$, formally:*

$$P(\updownarrow(i, j, \mathscr{G})) := \sum_{g \in \mathscr{W}} \prod_{e \in \mathscr{E}_g} P(e) \cdot \prod_{e \in \mathscr{E} \setminus \mathscr{E}_g} (1 - P(e)) \cdot \updownarrow(i, j, g),$$

where $\updownarrow(i, j, g)$ is an indicator function that returns one if there exists a path between nodes i and j in the (deterministic) possible graph g, and zero otherwise. For a given query node Q, the thesis' aim is to optimize the information gain, which is defined as the total weight of nodes reachable from Q.

Definition 20 (Expected Information Flow)**.** *Let $Q \in \mathscr{V}$ be a node and let $\mathscr{G} = (\mathscr{V}, \mathscr{E}, W, P)$ be a probabilistic graph, then $flow(Q, \mathscr{G})$ denotes the random variable of the sum of vertex weights of all nodes in V reachable from Q, formally:*

$$flow(Q, \mathscr{G}) := \sum_{v \in \mathscr{V}} P(\updownarrow(Q, v, \mathscr{G})) \cdot W(v).$$

Due to linearity of expectations, and exploiting that $W(v)$ is deterministic, we can compute the expectation $\mathbb{E}(flow(Q, \mathscr{G}))$ of this random variable as

$$\mathbb{E}(flow(Q, \mathscr{G})) = \mathbb{E}(\sum_{v \in \mathscr{V}} P(\updownarrow(Q, v, \mathscr{G})) \cdot W(v)) = \sum_{v \in \mathscr{V}} \mathbb{E}(P(\updownarrow(Q, v, \mathscr{G}))) \cdot W(v) \tag{5.2}$$

Given the definition of Expected Information Flow in Equation 5.2, the formal problem definition of optimizing the expected information flow of a probabilistic graph $\mathscr{G}$ for a constrained budget of edges can be stated.

Definition 21 (Maximum Expected Information Flow)**.** *Let $\mathscr{G} = (\mathscr{V}, \mathscr{E}, W, P)$ be a probabilistic graph, let $Q \in \mathscr{V}$ be a query node and let k be a non-negative integer. The Maximum Expected Information Flow*

$$MaxFlow(\mathscr{G}, Q, k) = argmax_{G=(\mathscr{V}, \mathscr{E}' \subseteq \mathscr{E}, W, P), |\mathscr{E}'| \leq k} \mathbb{E}(flow(Q, \mathscr{G})), \tag{5.3}$$

is the subgraph of $\mathscr{G}$ maximizing the information flow towards Q constrained to having at most k edges.

Computing MaxFlow$(\mathcal{G},Q,k)$ efficiently requires to overcome two NP-hard subproblems. First, the computation of the expected information flow $\mathbb{E}(\text{flow}(Q,\mathcal{G}))$ to vertex Q for a given probabilistic graph $\mathcal{G}$ is NP-hard as shown in [32]. In addition, the problem of selecting the optimal set of k vertices to maximize the information flow MaxFlow$(\mathcal{G},Q,k)$ is a NP-hard problem in itself, as shown in the following.

Theorem 2. *Even if the Expected Information Flow flow$(Q,\mathcal{G})$ to a vertex Q can be computed in $O(1)$ for any probabilistic graph $\mathcal{G}$, the problem of finding MaxFlow$(\mathcal{G},Q,k)$ is still NP-hard.*

Proof. In this proof, it is shown that a special case of computing MaxFlow$(\mathcal{G},Q,k)$ is *NP-complete*, thus, implying that the general problem is *NP-hard*. As shown in Figure 5.7, we reduce the *0-1 knapsack problem* to the problem of computing MaxFlow$(\mathcal{G},Q,k)$. Thus, assume a 0-1 knapsack problem: Given a capacity integer W and given a set $\{i_1,...,i_n\}$ of n items each having an integer weight w_i and an integer value v_i. The *0-1 knapsack* problem is to find the optimal vector $x=(x_1,...,x_n)\in\{0,1\}^n$ such that $\sum_{i=1}^{n} v_i\cdot x_i$, subject to $\sum_{i=1}^{n} w_i\cdot x_i\leq W$. This problem is known to be *NP-complete* [107]. We reduce this problem to the problem of computing MaxFlow$(\mathcal{G},Q,k)$ as follows. Let $\mathcal{G}=(\mathcal{V},\mathcal{E},G,P)$ be a probabilistic graph such that Q is connected to n nodes $\{V_1,...,V_n\}$ (one node for each item of the knapsack problem). Each node V_i is connected to a chain of w_i-1 nodes $\{V_i^1,...,V_i^{w_i-1}\}$. All edges have a probability of one, i.e., $P(v\in\mathcal{V})=1$. The information of a node is set to w_i if it is the (only) leaf node $v_i^{w_i-1}$ of the branch of $\mathcal{G}$ connected to V_i and zero otherwise. Finally, set $k=W$. Then, the solution of the *0-1 knapsack* problem can be derived from the constructed MaxFlow$(\mathcal{G},Q,k)$ problem by selecting all items n_i such that the corresponding node $v_i^{w_i-1}$ is connected to Q. Thus, if we can solve the MaxFlow$(\mathcal{G},Q,k)$ problem in polynomial time, then we can solve the *0-1 knapsack problem* in polynomial time: A contradiction assuming $P\neq NP$. □

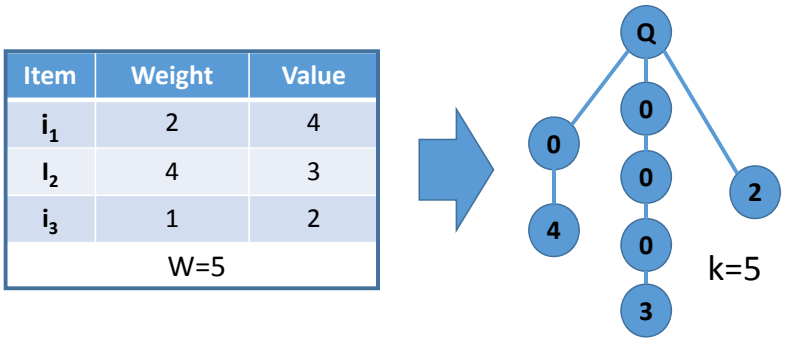


Figure 5.7: *Example of the Knapsack Reduction of Theorem 2*

5.2.4 Roadmap

To compute MaxFlow$(\mathcal{G}, Q, k)$, we first need an efficient solution to approximate the reachability probability $\mathbb{E}(P(\updownarrow(Q, v, \mathcal{G})))$ from Q to/from a single node v. Since this problem can be shown to be *#P-hard*, Section 5.2.5.3 presents an approximation technique which exploits stochastic independencies between branches of a spanning tree of subgraph G rooted at Q. This technique allows to aggregate independent subgraphs of G efficiently, while exploiting a sampling solution for components of the graph MaxFlow$(\mathcal{G}, Q, k)$ that contains cycles.

Once we can efficiently approximate the flow $\mathbb{E}(P(\updownarrow(Q, v, \mathcal{G})))$ from Q to each node $v \in \mathcal{V}$, we next tackle the problem of efficiently finding a subgraph MaxFlow$(\mathcal{G}, Q, k)$ that yields a near-optimal expected information flow given a budget of k edges in Section 5.2.6. Due to the theoretic result of Theorem 2, the thesis proposes heuristics to choose k edges from $\mathcal{G}$. Finally, the experiments in Section 5.2.7 support the theoretical intuition that the thesis' solutions for the two aforementioned subproblems synergize: An optimal subgraph will choose a budget of k edges in a tree-like fashion, to reach large parts of the probabilistic graph. At the same time, the solutions exploit tree-like subgraphs for efficient probability computation.

5.2.5 Expected Flow Estimation via Flow Tree

In this section, the expected information flow of a given subgraph $G \subseteq \mathcal{G}$ is estimated. Following Equation 5.2, the reachability probability $P(\updownarrow(Q, v, \mathcal{G}))$ between Q and a node v can be used to compute the total expected information flow $\mathbb{E}(\text{flow}(Q, \mathcal{G}))$. This problem of computing the reachability probability between two nodes has been shown to be *#P-hard* [72, 32] and sampling solutions have been proposed to approximate it [130, 52]. In this section, the thesis presents a solution to identify subgraphs of $\mathcal{G}$ for which the information can be computed analytically and efficiently, such that expensive numeric sampling only has to be applied to small subgraphs. First, the concept of Monte-Carlo sampling of a subgraph is introduced.

5.2.5.1 Traditional Monte-Carlo Sampling

Lemma 1. *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W, P)$ be an uncertain graph and let $\mathcal{S}$ be a set of sample worlds drawn randomly and unbiased from the set $\mathcal{W}$ of possible graphs of $\mathcal{G}$. Then the average information flow in samples in $\mathcal{S}$*

$$\frac{1}{|\mathcal{S}|} \sum_{g \in \mathcal{S}} \mathit{flow}(Q, g) = \frac{1}{|\mathcal{S}|} \cdot \sum_{g \in \mathcal{S}} \sum_{v} \updownarrow(Q, v, g) \cdot W(v) \tag{5.4}$$

is an unbiased estimator of the expected information flow $\mathbb{E}(\mathit{flow}(Q, \mathcal{G}))$, where $\updownarrow(Q, v, g)$ is an indicator function that returns one if there exists a path between nodes Q and v in the (deterministic) sample graph g, and zero otherwise.

Proof. For μ to be an unbiased estimator of $\mathbb{E}(\text{flow}(Q,\mathcal{G}))$, we have to show that $\mathbb{E}(\mu) = \mathbb{E}(\text{flow}(Q,\mathcal{G}))$. Substituting μ yields $\mathbb{E}(\mu) = \mathbb{E}(\frac{1}{|\mathcal{S}|}\sum_{g\in\mathcal{S}}\text{flow}(Q,g))$. Due to linearity of expectations, this is equal to $\frac{1}{|\mathcal{S}|}\sum_{g\in\mathcal{S}}\mathbb{E}(\text{flow}(Q,g))$. The sum over $|\mathcal{S}|$ identical values can be replaced by a factor of $|\mathcal{S}|$. Reducing this factor yields $\mathbb{E}(\text{flow}(Q,g\in\mathcal{S}))$. Following the assumption of unbiased sampling $\mathcal{S}$ from the set $\mathcal{W}$ of possible worlds, the expected information flow $\mathbb{E}(\text{flow}(Q,g))$ of a sample possible world $g\in\mathcal{S}$ is equal to the expected information flow $\mathbb{E}(\text{flow}(Q,\mathcal{G}))$. □

Naive sampling of the whole graph $\mathcal{G}$ has disadvantages: First, this approach requires to compute reachability queries on a set of possibly large sampled graphs. Second, a rather large approximation error is incurred. The thesis approaches these drawbacks by first describing how non-cyclic subgraphs, i.e., trees (cf. Definition 4), can be processed in order to compute the information flow exactly and efficiently without sampling. For cyclic subgraphs, the thesis shows how sampled information flows can be used to compute the information flow in the full graph.

5.2.5.2 Mono-Connected vs. Bi-Connected graphs

The main observation that will be exploited is the following: if there exists only one possible path between two vertices, then we can compute their reachability probability efficiently.

Definition 22 (Mono-Connected Nodes)**.** *Let $\mathcal{G}=(\mathcal{V},\mathcal{E},W,P)$ be a probabilistic graph and let $A,B\in V$. If $path(A,B)=(A=v_0,v_1,...,v_{k-1},v_k=B)$ is* the only *path between A and B, i.e., there exists no other path $p\in\mathcal{V}\times\mathcal{V}\times\mathcal{V}^*$ that satisfies Definition 18, then we denote A and B as* mono-connected*.*

In the following, when the query vertex Q is clear from the context, a vertex A is called mono-connected if it is mono-connected to the query vertex Q.

Lemma 2. *If two vertices A and B are mono-connected in a probabilistic graph $\mathcal{G}$, then the reachability probability between A and B is equal to the product of the edge probabilities included in path(A,B), i.e.,*

$$\updownarrow(A,B,\mathcal{G}) = \prod_{i=0}^{k-1} P((v_i,v_{i+1})) \text{ with } v_i \in path(A,B)$$

Proof. Following possible world semantics as defined in Definition 19, the reachability probability $\updownarrow(A,B,\mathcal{G})$ is the sum of probabilities of all possible worlds where B is connected to A. It is shown that A and B are connected in a possible graph g iff all $k-1$ edges $e_i=(v_i,v_{i+1})$ with $v_i,v_{i+1}\in\text{path}(A,B)$ exist.
$\Rightarrow$: By contradiction: Let A and B be connected in g, and let any edge on $\text{path}(A,B)$ be missing. Then there must exist a path $\text{path}^{prime}(A,B)\neq\text{path}(A,B)$ which contradicts the assumption that A and B are mono-connected.

$\Leftarrow$: If all edges on path(A,B) exist, then B is connected to A following the assumption that path(A,B) is a path from A to B.

Due to the assumption of independent edges, the probability that all edges in path(A,B) exist is given by $\prod_{i=0}^{k-1} P((v_i,v_{i+1}))$. □

Definition 23 (Mono-Connected Graph)**.** *A probabilistic graph $\mathcal{G} = (\mathcal{V},\mathcal{E},W,P)$ is called mono-connected, iff all pairs of vertices in $\mathcal{V}$ are mono-connected.*

Next, Lemma 2 is generalized to whole subgraphs, such that a specified vertex Q in that subgraph has a unique path to all other vertices in the subgraph. Using Lemma 2, we constitute the following theorem that will be exploited in the remainder of this thesis.

Theorem 3. *Let $\mathcal{G} = (\mathcal{V},\mathcal{E},G,P)$ be a probabilistic graph, let $Q \in \mathcal{V}$ be a node. If $\mathcal{G}$ is mono-connected, then $\mathbb{E}(flow(Q,\mathcal{G}))$ can be computed efficiently.*

Proof. $\mathbb{E}(\text{flow}(Q,\mathcal{G}))$ is the sum of reachability probabilities of all nodes, according to Equation 5.2. If $\mathcal{G}$ is connected and non-cyclic, we can guarantee that each node has exactly one path to Q and, thus, is mono-connected. Thus, Lemma 2 is applicable to compute the reachability probability between Q and each node $v \in \mathcal{V}$. Due to linearity of expectations, i.e., $\mathbb{E}(X+Y) = \mathbb{E}(X) + \mathbb{E}(Y)$ for random variables X and Y, we can aggregate individual reachability expectations, yielding $\mathbb{E}(\text{flow}(Q,\mathcal{G}))$. □

Analogously to Definition 22, bi-connected nodes are defined.

Definition 24 (Bi-Connected Nodes)**.** *Let $\mathcal{G} = (\mathcal{V},\mathcal{E},W,P)$ be a probabilistic graph and let $A,B \in \mathcal{V}$. If there exists (at least) two paths $path_1(A,B)$ and $path_2(A,B)$, such that $path_1(A,B) \neq path_2(A,B)$, then we denote A and B as* bi-connected*.*

Definition 25 (Bi-Connected Graph)**.** *A bi-connected graph [199, 90] is a connected probabilistic graph $\mathcal{G} = (\mathcal{V},\mathcal{E},W,P)$ such that removal of any one vertex $A \in \mathcal{V}$ will still yield a connected probabilistic graph.*

Lemma 3. *In a bi-connected graph $\mathcal{G}$ of size $|\mathcal{V}| \geq 3$, all pairs of vertices are bi-connected following Definition 24.*

Proof. By contradiction, let A,B be two nodes in $\mathcal{G}$ that are mono-connected. Let path(A,B) be the only path between them.
Case 1: path$(A,B) = (A,B)$ contains no other vertices: Since $\mathcal{G}$ is bi-connected, removal of vertex A yields a graph where B and C are connected by some path path(C,B). At the same time, removal of vertex B yields a graph where A and C are connected by some path path(A,C). Thus, the concatenation of these paths yields an alternative path between A and B, contradicting the assumption that (A,B) are mono-connected by path (A,B).

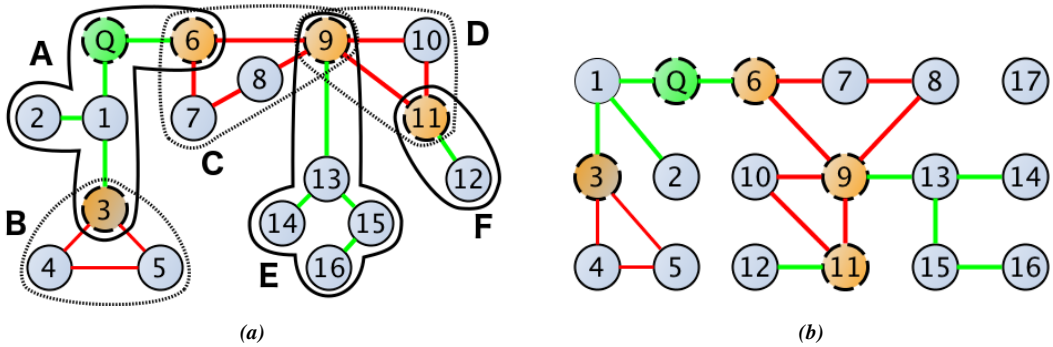


Figure 5.8: *Running example graph with corresponding* F-tree

Case 2: path$(A,B) = (A,C_1,...,C_n,B)$ contains other vertices. Let C_1 be such a vertex. Since $\mathcal{G}$ is bi-connected, removal of vertex C_1 yields a graph where A and B are still connected, contradicting the assumption that A and B are mono-connected by path(A,B) only. □

The information flow within a bi-connected graph can not be computed efficiently using Theorem 3, as the flow between any two nodes A to B is shared by more than one path. In the next section, the thesis proposes techniques to substitute bi-connected subgraphs by super-nodes, for which the information flow can be estimated using Monte-Carlo sampling exploiting Lemma 1. By substituting the bi-connected subgraphs by super-nodes for which the method applies sampling and memoizes the sampling information for these super-nodes, the procedure yields a mono-connected graph that uses the substituted super-nodes. This approach maximizes the partitions of the graph for which expensive Monte-Carlo estimation can be replaced using Theorem 3.

The next section will show how to achieve this goal, by employing a *F-tree* of the graph. This data structure borrowed from graph theory partitions the graph into bi-connected components (a.k.a. "blocks") generated by bi-connected subgraphs, and identifies vertices of the graph as *articulation vertices* to connect two bi-connected components. These articulation vertices are exploited by having them represent all the information flow that is estimated to flow to them from their corresponding bi-connected component.

5.2.5.3 Flow tree (F-tree)

In this section, it is proposed to adapt the block-cut tree [199, 90, 222] to partition a graph into independent bi-connected components. Instead of sampling the whole uncertain graph, the purpose of this index structure is to exploit Theorem 3 for mono-connected components, and to apply local Monte-Carlo within bi-connected components only. The employed *Flow tree* (F-tree) memoizes the information flow at each node. Before it is shown how to utilize the *F-tree* for efficient information flow computation, a formal definition is given.

Definition 26 (Flow tree)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W, P)$ be a probabilistic graph and let $Q \in \mathcal{V}$ be a vertex for which the expected information flow is computed. A* Flow tree *(*F-tree*) is a tree structure defined as follows:*
1) *Each component of the* F-tree *is a* connected subgraph *of $\mathcal{G}$. A component can be* mono-connected *or* bi-connected*.*
2) *A mono-connected component $MC = (MC.V \subseteq \mathcal{V}, MC.AV \in \mathcal{V})$ is a set of vertices $MC.V \cup MC.AV$ that form a mono-connected subgraph (cf. Definition 23) in $\mathcal{G}$. The vertex $MC.AV$ is called* articulation vertex*. Intuitively, a mono-connected components represents a tree-like structure rooted in $MC.AV$. Using Theorem 3, we can efficiently compute the information flow from all vertices $MC.V$ to $MC.AV$.*
3) *A bi-connected component $BC = (BC.V, BC.P(v),\ BC.AV)$ is a set of vertices $BC.V \cup BC.AV$ of size greater than two that form a bi-connected subgraph $\mathcal{G}'$ in $\mathcal{G}$ according to Definition 25. Intuitively, a bi-connected component represents a subgraph describing a cycle. In this case, we can estimate the likelihood of being connected to the articulation vertex $BC.AV$ using Monte-Carlo sampling in Lemma 1. The function $BC.P(v) : BC.V \mapsto [0,1]$ maps each vertex $v \in BC.V$ to the estimated reachability probability $reach(v, BC.AV)$ of v being connected to $BC.AV$ in $\mathcal{G}$.*
4) *For each pair of (mono- or bi-connected) components (C_1, C_2), it holds that the intersection $C_1.V \cap C_2.V = \emptyset$ of vertices is empty. Thus, each vertex in $\mathcal{V}$ is mapped to at most one component's vertex set.*
5) *Two different components may share the same articulation vertex, and the articulation vertex of one component may be in the vertex set of another component.*
6) *The articulation vertex of the root of an* F-tree *is $Q \in \mathcal{V}$.*

Intuitively speaking, a component is a set of vertices together with an articulation vertex that all information must flow through in order to reach Q. By the iterative construction algorithm presented in Section 5.2.5.4, each component is guaranteed to have such an articulation vertex, guiding the direction to vertex Q. The idea of the *F-tree* is to use components as virtual nodes, such that all actual vertices of a component send their information to their articulation vertex. Then the articulation vertex forwards all information to the next component, until the root of the tree is reached where all information is sent to articulation vertex Q.

Example 2. *As an example for an* F-tree*, consider Figure 5.8a, showing a probabilistic graph. For brevity, assume that each edge $e \in \mathcal{E}$ has an existential probability of $p(e) = 0.5$ and that all vertices $v \in \mathcal{V}$ have an information weight corresponding to their id, e.g., vertex* 6 *has a weight of six. A corresponding* F-tree *is shown in Figure 5.8b. A mono-connected component is given by $A = (\{1,2,3,6\}, Q)$. For this component, we can exploit Theorem 3 to analytically compute the flow of information from any vertex in $\{1,2,3,6\}$ to articulation vertex Q: vertices* 3 *and* 6 *are connected to Q with probability* 0.5*. Thus, these nodes contributed an expected information flow of $3 \cdot 0.5 = 1.5$ and $6 \cdot 0.5 = 3$, respectively. Vertices* 2 *and* 3 *are connected to Q with a probability of $0.5 \cdot 0.5 = 0.25$, respectively, following Lemma 2. Thus, these nodes contribute*

an expected information of $2 \cdot 0.25 = 0.5$ *and* $3 \cdot 0.25 = 0.75$. *Following Theorem 3, we can aggregate these probabilities to obtain the expected information flow from vertices* $\{1,2,3,6\}$ *to articulation vertex Q as* 5.75.

A bi-connected component is defined by $B = (\{4,5\},3)$, *representing a sub-graph having a cycle. Having a cycle, we cannot exploit Theorem 3 to compute the flow of a vertex in* $\{4,5\}$ *to vertex* 3. *But we can sample the subgraph spanned by vertices in* $\{3,4,5\}$ *to estimate probabilities that vertices* $\{4,5\}$ *are connected to articulation vertex* 3 *using Lemma 1. With sufficient samples, this will yield a probability of around* 0.375 *for both vertices to be reached. Again using Theorem 3, we compute an information flow of* $0.375 \cdot 4 + 0.375 \cdot 5 = 3.375$ *to articulation vertex* 3. *Given this expected flow, we can use the mono-connected component A to compute the expected information analytically that is further propagated from the articulation vertex* 3 *of component B to the articulation vertex Q of A. As the articulation vertex of component B is in the vertex set of component A, component B is a child of component A in Figure 5.8b since B propagates its information to A. As we have already computed above, the probability of vertex* 3 *to be connected to its articulation vertex Q is* 0.25, *yielding an information flow worth* $3.375 \cdot 0.25 = 0.84375$ *units flowing from vertices* $\{4,5\}$ *to Q. Again, exploiting Theorem 3, we can aggregate this to a total flow of* $5.75 + 0.84375 = 6.59375$ *from vertices {1,2,3,4,5,6} to Q.*

Another bi-connected component is $C = (\{7,8,9\},6)$, *for which we can estimate the information flow from vertices* 7, 8, *and* 9 *to articulation vertex* 6 *numerically using Monte-Carlo sampling. Since vertex* 6 *is in A, component C is a child of A. We find another bi-connected component* $D = (\{10,11\},9)$, *and two more mono-connected components* $E = (\{13,14,15,16\},9)$ *and* $F = (\{12\},11)$.

In this example, the structure of the *F-tree* allows to compute or approximate the expected information flow to Q from each vertex. For this purpose, only three small components B, C and D need to be sampled. This is a vast reduction of sampling space compared to a naive Monte-Carlo approach that samples the full graph: rather than sampling a single random variable having $2^{|\mathscr{E}|} = 2^{19} = 524288$ possible worlds, we only need to sample three random variables corresponding to the bi-connected components B, C and D having $2^3 = 8$, $2^4 = 16$, and $2^3 = 8$ possible worlds, respectively. Clearly, this approach reduces the number of edges (marked in red in Figure 5.8a) that need to be sampled in each iteration. More importantly, the experiments show that this approach of sampling components independently vastly decreases the variance of the total information flow yielding a more precise estimation at the same number of samples.

Having defined syntax and semantics of the *F-tree*, the next section shows how to maintain the structure of an *F-tree* when additional edges are selected. It is important to note that the intention is not to insert all edges of a probabilistic graph $\mathscr{G}$ into the *F-tree*. Rather, we only add the edges that are selected to compute the maximum flow $\text{MaxFlow}(\mathscr{G},Q,k)$ given a constrained budget of k edges. Thus, even in a case where all vertices a bi-connected, such as in the initial example in Figure 5.6a, it is notable and supported by the experimental evaluation, that an

optimal selection of edges prefers a spanning-tree-like topology, which synergizes well with the definition of an *F-tree*. The next section shows how to build the structure of the *F-tree* iteratively by adding edges to an initially empty graph.

The next subsection proposes an algorithm, to update an *F-tree* when a new edge is selected, starting at a trivial *F-tree* that contains only one component $(\emptyset, Q)$. Using this edge-insertion algorithm, the thesis shows how to choose promising edges to be inserted to maximize the expected information flow. The selection of the edges of the *F-tree* will be shown in section 5.2.6.

5.2.5.4 Insertion of Edges into an F-tree

Following Definition 26 of an *F-tree*, each vertex $v \in \mathcal{G}$ is assigned to either a single mono-connected component (noted by a flag *v.isMC* in the algorithm below), a single bi-connected component (noted by *v.isBC*), or to no component, and thus disconnected from Q, noted by *v.isNew*. To insert a new edge (v_{src}, v_{dest}), the edge-insertion algorithm derived in this section differs between these cases as follows:
Case I) $v_{src}.isNew$ and $v_{dest}.isNew$: We omit this case, as the edge selection algorithms presented in Section 5.2.6 always ensure a single connected component and initially the *F-tree* contains only vertex Q.
Case II) $v_{src}.isNew$ exclusive-or $v_{dest}.isNew$: Due to considering undirected edges, we assume without loss of generality that $v_{dest}.isNew$. Thus, v_{src} is already connected to *F-tree*.

Case IIa): $v_{src}.isMC$: In this case, a new dead end is added to the mono-connected structure MC_{src} which is guaranteed to remain mono-connected. We add v_{dest} to $MC_{src}.V$.

Case IIb): $v_{src}.isBC$: In this case, a new dead end is added to the bi-connected structure BC_{src}. This dead end becomes a new mono-connected component $MC = (\{v_{dest}\}, v_{src})$. Intuitively speaking, we know that vertex v_{dest} has no other choice but propagating its information to v_{src}. Thus, v_{src} becomes the articulation vertex of MC. The bi-connected component BC_{src} adds the new mono-connected component MC to its list of children.
Case III) v_{src} and v_{dest} belong to the same component, i.e., $C_{src} = C_{dest}$

Case IIIa) This component is a bi-connected component BC: Adding a new edge between v_{src} and v_{dest} within component BC may change the reachability $BC.P(v)$ of each vertex $v \in BC.V$ to reach their articulation vertex $BC.AV$. Therefore, BC needs to be re-sampled to numerically estimate the reachability probability function $P(v)$ for each $v \in BC.V$.

Case IIIb): This component is a mono-connected component MC: In this case, a new cycle is created within a mono-connected component, thus, some vertices within MC may become bi-connected. We need to (i) identify the set of vertices affected by this cycle, (ii) split these vertices into a new bi-connected component, and (iii) handle the set of vertices that have been disconnected from MC by the new cycle. These three steps are performed by the *splitTree*(MC, v_{src}, v_{dest}) function as follows: (i) The new cycle is identified as follows: Compare the (unique) paths of v_{src} and v_{dest} to $MC.AV$, and find the first vertex $v_\wedge$ that appears in both

paths. Now we know that the new cycle is decribed by $path(v_\wedge, v_{src}), path(v_{dest}, v_\wedge)$ and the new edge between v_{src} and v_{dest}. (ii) All of these vertices are added to a bi-connected component $BC = (path(v_\wedge, v_{src}) \cup path(v_{dest}, v_\wedge) \setminus v_\wedge, P(v), v_\wedge)$ using $v_\wedge$ as their articulation vertex. All vertices in MC having $v_\wedge$ (except $v_\wedge$ itself) on their path are removed from MC. The probability mass function $P(v)$ is estimated by sampling the subgraph of vertices in $BC.V$. (iii) Finally, orphans of MC that have been split off from MC due to the creation of BC need to be collected into new mono-connected components. Such orphans having a vertex of the cycle BC on their path to $MC.AV$ will be grouped by these vertices: For each $v_i \in BC.V$, let $orphan_i$ denote the set of orphans separated by v_i (separated means v_i being the first vertex in $BC.V$ on the path to $MC.AV$). For each such group, a new mono-connected component $MC_i = (orphan_i, v_i)$ is created. All these new mono-connected components with $v_i \in BC.V$ become children of BC. If $MC.V$ is now empty, thus, all vertices of MC have been reassigned to other components, then MC is deleted and BC will be appended to the list of children of the component C where $BC.AV = v_\wedge \in C.V$. In case of $MC.V$ being not empty, we are left over with a mono-connected component MC with $v_\wedge \in MC.V$. The new bi-connected component BC becomes a child of MC.

Case IV) v_{src} and v_{dest} belong to different components $C_{src} \neq C_{dest}$. Since the *F-tree* is a tree-structure itself, we can identify the lowest common ancestor C_{anc} of C_{src} and C_{dest}. The insertion of edge (v_{src}, v_{dest}) has incurred a new cycle $\bigcirc$ going from C_{anc} to C_{src}, then to C_{dest} via the new edge, and then back to C_{anc}. This cycle may cross mono-connected and bi-connected components, which all have to be adjusted to account for the new cycle. We need to identify all vertices involved to create a new cyclic, thus bi-connected, component for $\bigcirc$, and we need to identify which parts remain mono-connected. In the following cases, all components involved in $\bigcirc$ are adjusted iteratively. First, we initialize $\bigcirc = (\emptyset, P, v_{anc})$, where v_{anc} is the vertex within C_{anc} where the cycle meets if C_{anc} is a mono-connected component, and $C_{anc}.AV$ otherwise. Let C denote the component that is currently adjusted:

Case IVa) $C = C_{anc}$: In this case, the new cycle may enter C_{anc} from two different articulation vertices. In this case, *Case III* is applied, treating these two vertices as v_{src} and v_{dest}, as these two vertices have become connected transitively via the big cycle $\bigcirc$.

Case IVb) C is a bi-connected component: In this case C becomes absorbed by the new cyclic component $\bigcirc$, thus $\bigcirc.V = \bigcirc.V \cup C.V$, and $\bigcirc$ inherits all children from C. The rational is that all vertices within C are able to access the new cycle.

Case IVc) C is a mono-connected component: In this case, one path in C from one vertex v to $C.AV$ is now involved in a cycle. All vertices involved in $path(v, C.AV)$ are added to $\bigcirc.V$ and removed from C. The operation $splitTree(C, v, C.AV)$ is called to create new mono-connected components that have been split off from C and become connected to $\bigcirc$ via their individual articulation vertices.

In the following, the graph of Figure 5.8a and its corresponding F-tree representation of Figure 5.8a is used to insert additional edges and to illustrate the interesting cases of the insertion algorithm of Section 5.2.5.4.

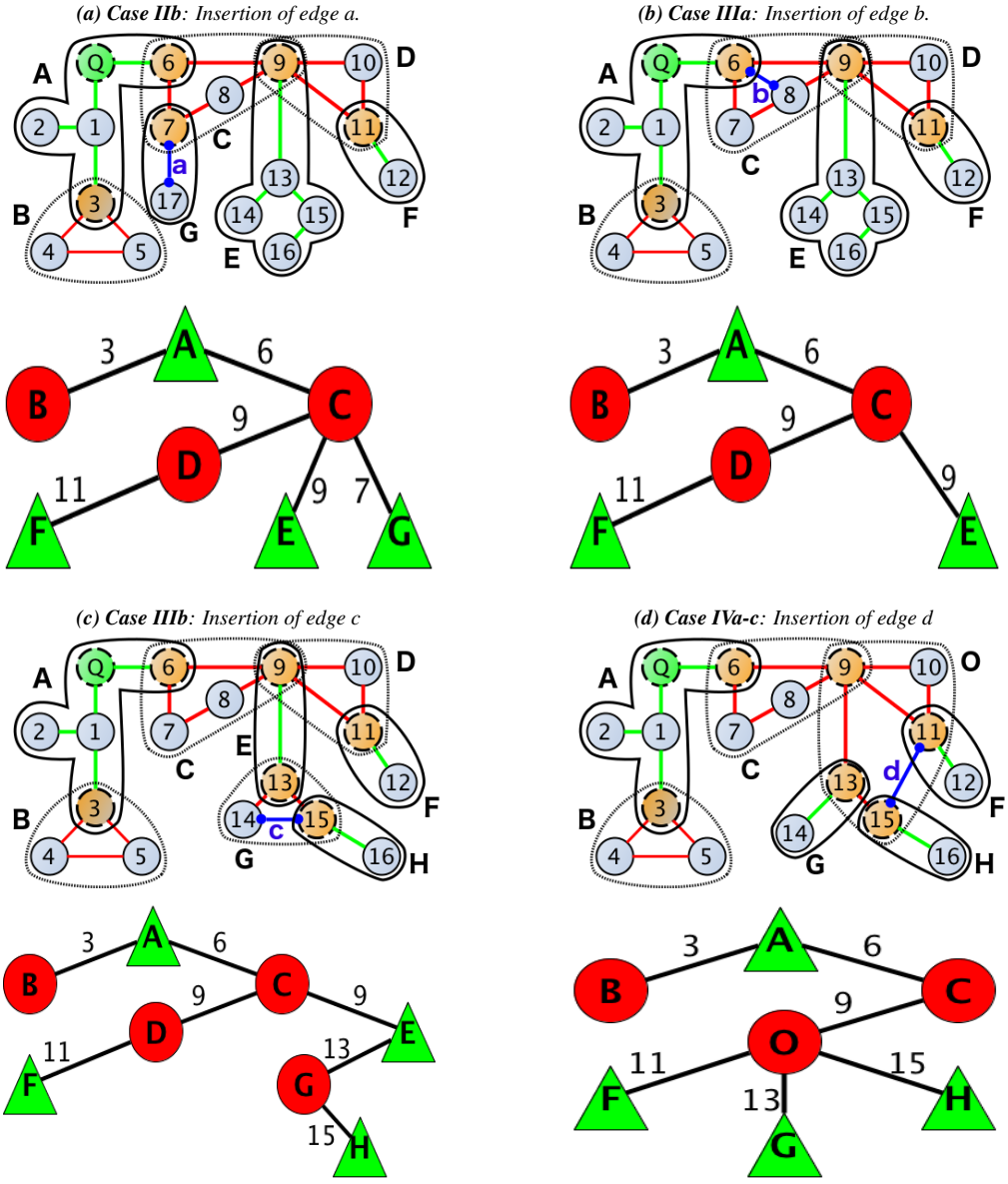


Figure 5.9: *Examples of edge insertions and* F*-tree update cases using the running example of Figure 5.8a.*

5.2.5.5 Insertion Examples

In the following, the graph of Figure 5.8a and its corresponding F-tree (FT) representation of Figure 5.8b is used to insert additional edges and to illustrate the interesting cases of the insertion algorithm of Section 5.2.5.4.

Let us start by an example for **Case II** in Figure 5.9a. Here, we insert a new edge $a = (7, 17)$, thus connecting a new vertex 17 to the *FT*. Since vertex 7 belongs to the bi-connected component *BC*, we apply **Case IIb**. A new mono-connected component $G = (\{17\}, 7)$ is created and added to the children of *BC*.

In Figure 5.9b, we insert a new edge $b = (6, 8)$ instead. In this case, the two connected vertices are already part of the *FT*, thus, Case II does not apply. We find that both vertices belong to the same component C. Thus, **Case III** is used and more specifically, since component C is a bi-connected component *BC*, **Case IIIa** is applied. In this case, no components need to be changed, but the probability function $BC.P(v)$ has to re-approximated, as the probabilities of nodes 7, 8 and 9 will have increased probability of being connected to articulation vertex 6, due to the existence of new paths arising by inserting edge b.

Next, in Figure 5.9c, an edge is inserted between vertices 14 and 15. Both vertices belong to the mono-connected component E, thus, **Case IIIb** is applied here. After insertion of edge c, the previously mono-connected component $E = (\{13, 14, 15, 16\}, 9)$ now contains a cycle involving vertices 13, 14 and 15. (i) We identify this cycle by considering the previous paths from vertices 14 and 15 to their articulation vertex 9. These paths are $(14, 13, 9)$ and $(15, 13, 9)$, respectively. The first common vertex on this path is 13; (ii) We create a new bi-connected component $G = (\{14, 15\}, 13)$, containing all vertices of this cycle using the first common vertex 13 as articulation vertex. We further remove these vertices except the articulation vertex 13 from the mono-connected component E; the probability function $G.P(v)$ is initialized by sampling the reachability probabilities within G; and component G is added to the list of children of E; (iii) Finally, orphans need to be collected. These are vertices in E, which have now become bi-connected to Q, because their (previously unique) path to their former articulation vertex 9 crosses a new cycle. We find that one vertex, vertex 16, had 15 as the first removed vertex on its path to 9. Thus, vertex 16 is moved from component E into a new mono-connected component $H = (\{16\}, 15)$, terminating this case. Summarizing, vertex 16 in component H now reports its information flow to vertex 15 in component G, for which the information flow to articulation vertex 13 in component G is approximated using Monte-Carlo sampling. This information is then propagated analytically to vertex 9 in component E, subsequently, the remaining flow that has been propagated all this way, is approximatively propagated to articulation vertex 6 in component C, which allows to analytically compute the flow to articulation vertex Q.

For the last case, **Case IV**, considering Figure 5.9d, where a new edge $d = (11, 15)$ connected two vertices belonging to two different components D and E. We start by identifying the cycle that has been created within the *FT*, involving components D and E, and meeting at the first

common ancestor component C. For each of these components in the cycle (D, C, E), one of the sub-cases of Case IV is used. For component C, we have that $C = C_{anc}$ is the common ancestor component, thus, triggering **Case IVa**. We find that both components D and E used vertex 9 as their articulation vertex v_{anc}. Thus, the only cycle incurred in component C is the (trivial) cycle (9) from vertex 9 to itself, which does not require any action. We initialize the new bi-connected component $\bigcirc = (\emptyset, \perp, 9)$, which initially holds no vertices, and has no probability mass function computed yet (the operator $\perp$ can be read as null or not-defined) and uses $v_{anc} = 9$ as articulation vertex. For component D, we apply **Case IVb**, as D is a bi-connected component, it becomes absorbed by a new bi-connected component $\bigcirc$, now having $\bigcirc = (\{10, 11\}, \perp, 9)$. For the mono-connected component E **Case IVc** is used. We identify the path within E that is now involved in a cycle, by using the path $(15, 13, 9)$ between the involved vertex 15 to articulation vertex 9. All nodes on this path are added to $\bigcirc$, now having $\bigcirc = (\{10, 11, 15, 13\}, \perp, 9)$. Using the $splitTree(\cdot)$ operation similar to Case III, we collect orphans into new mono-connected components, creating $G = (\{14\}, 13)$ and $H = (\{16\}, 15)$ as children of $\bigcirc$. Finally, Monte-Carlo sampling is used to approximate the probability mass function $\bigcirc.P(v)$ for each $v \in \bigcirc.V$.

5.2.6 Optimal Edge Selection

The previous section presented the *F-tree*, a data structure to compute the expected information flow in a probabilistic graph. Based on this structure, heuristics to find a near-optimal set of k edges maximizing the information flow $\text{MaxFlow}(\mathcal{G}, Q, k)$ to a vertex Q (see Definition 21) are presented in this section. Therefore, the thesis first presents a *Greedy*-heuristic to iteratively add the locally most promising edges to the current result. Based on this *Greedy* approach, the thesis presents improvements, aiming at minimizing the processing cost while maximizing the expected information flow.

5.2.6.1 Greedy Algorithm

Aiming to select edges incrementally, the *Greedy* algorithm initially uses the probabilistic graph $G_0 = (\mathcal{V}, \mathcal{E}_0 = \emptyset, P)$, which contains no edges. In each iteration i, a set of candidate edges *candList* is maintained, which contains all edges that are connected to Q in the current graph G_i, but which are not yet selected in $\mathcal{E}_i$. Then, each iteration selects an edge e in addition maximizing the information flow to Q, such that $G_{i+1} = (\mathcal{V}, \mathcal{E}_i \cap e, P)$, where

$$e = \underset{e \in candList}{\text{argmax}}\ \mathbb{E}(\text{flow}(Q, (\mathcal{V}, \mathcal{E}_i \cap e, P))). \tag{5.5}$$

For this purpose, each edge $e \in candList$ is probed, by inserting it into the current *F-tree* using the insertion method presented in Section 5.2.5.3. Then, the gain in information flow incurred by this insertion is estimated by Equation 1. After k iterations, the graph $G_k = (\mathcal{V}, \mathcal{E}_k, P)$ is returned.

5.2.6.2 Component Memoization

This section introduces an optimization reducing the number of computations for bi-connected components for which their reachability probabilities have to be estimated using Monte-Carlo sampling, by exploiting stochastic independence between different components in the *F-tree*. During each *Greedy*-iteration, a whole set of edges *candList* is probed for insertion. Some of these insertions may yield new cycles in the *F-tree*, resulting from cases III and IV. Using component *Memoization*, the algorithm memoizes, for each edge e in *candList*, the probability mass function of any bi-connected component *BC* that had to be sampled during the last probing of e. Should e again be inserted in a later iteration, the algorithm checks if the component has changed, in terms of vertices within that component or in terms of other edges that have been inserted into that component. If the component has remained unchanged, the sampling step is skipped, using the memoized estimated probability mass function instead.

5.2.6.3 Sampling Confidence Intervals

A Monte Carlo sampling is controlled by a parameter *samplesize* which corresponds to the number of samples taken to approximate the information flow of a bi-connected component to its articulation vertex. In each iteration, the amount of samples is reduced by introducing confidence intervals for the information flow for each edge $e \in$ *candList* that is probed. The idea is to prune the sampling of any probed edge e for which we can conclude that, at a sufficiently large level of significance α, there must exist another edge $e' \neq e$ in *candList*, such that e' is guaranteed to have a higher information flow than e, based on the current number of samples only. To generate these confidence intervals, we recall that, following Equation 5.4 the expected information flow to Q is the sample-average of the sum of information flow of each individual vertex. For each vertex v, the random event of being connected to Q in a random possible world follows a binomial distribution, with an unknown success probability p. To estimate p, given a number S of samples and a number $0 \leq s \leq S$ of 'successful' samples in which Q is reachable from v, the model borrows techniques from statistics to obtain a two sided $1-\alpha$ confidence interval of the true probability p. A simple way of obtaining such confidence interval is by applying the *Central Limit Theorem* of Statistics to approximate a binomial distribution by a normal distribution.

Definition 27 (α-Significant Confidence Interval)**.** *Let S be a set of possible graphs drawn from the probabilistic graph $\mathcal{G}$, and let $\hat{p} := \frac{s}{S}$ be the fraction of possible graphs in S in which Q is reachable from v. With a likelihood of $1-\alpha$, the true probability $\mathbb{E}(P(\updownarrow(Q,v,\mathcal{G})))$ that Q is reachable from v in the probabilistic graph G is in the interval*

$$\hat{p} \pm z \cdot \sqrt{\hat{p}(1-\hat{p})}, \tag{5.6}$$

where z is the $100 \cdot (1-0.5 \cdot \alpha)$ percentile of the standard normal distribution. We denote the lower bound as $\mathbb{E}_{lb}(P(\updownarrow(Q,v,\mathcal{G})))$ and the upper bound as $\mathbb{E}_{ub}(P(\updownarrow(Q,v,\mathcal{G})))$. We use $\alpha = 0.01$.

To obtain a lower bound of the expected information flow to Q in a graph $\mathscr{G}$, we use the sum of lower bound flows of each vertex using Equation 5.4 to obtain

$$\mathbb{E}_{lb}(\text{flow}(Q,\mathscr{G})) = \sum_{v \in \mathscr{V}} \mathbb{E}_{lb}(P(\updownarrow(Q,v,\mathscr{G}))) \cdot W(v)$$

as well as the upper bound

$$\mathbb{E}_{ub}(\text{flow}(Q,\mathscr{G})) = \sum_{v \in \mathscr{V}} \mathbb{E}_{ub}(P(\updownarrow(Q,v,\mathscr{G}))) \cdot W(v)$$

Now, at any iteration i of the *Greedy* algorithm, for any candidate edge $e\prime \in candList$ having an information flow lower bounded by $lb := \mathbb{E}_{lb}(\text{flow}(Q,\mathscr{G}_i) \cup e)$, any other candidate edge $e' \in candList$ is pruned having an upper bound $ub := \mathbb{E}_{ub}(\text{flow}(Q,\mathscr{G}_i \cup e'))$ iff $lb > ub$. The rational of this pruning is that, with a confidence of $1-\alpha$, it is guaranteed that inserting $e\prime$ yields less information gain than inserting e. To ensure that the *Central Limit Theorem* is applicable, this pruning step is only applied if at least 30 sample worlds have been drawn for both probabilistic graphs.

5.2.6.4 Delayed Sampling

For the last heuristic, the number of Monte-Carlo samplings is reduced that need to be performed in each iteration of the *Greedy* Algorithm in Section 5.2.6.1. In a nutshell, the idea is that an edge, which yields a much lower information gain than the chosen edge, is unlikely to become the edge having the highest information gain in the next iteration. For this purpose, the thesis introduces a delayed sampling heuristic. In any iteration i of the *Greedy* Algorithm, let e denote the best selected edge, as defined in Equation 5.5. For any other edge $e' \in candList$, let us define its potential $pot(e') := \frac{\mathbb{E}(\text{flow}(Q,(\mathscr{V},\mathscr{E}_i \cap e',P))}{\mathbb{E}(\text{flow}(Q,(\mathscr{V},\mathscr{E}_i \cap e,P))}$, as the fraction of information gained by adding edge $e\prime$ compared to the best edge e which has been selected in an iteration. Furthermore, let us define the cost $cost(e')$ as the number of edges that need to be sampled to estimate the information gain incurred by adding edge e'. If the insertion of e' does not incur any new cycles, then $cost(e')$ is zero. Now, after iteration i where edge e' has been probed but not selected, we define a sampling delay

$$d(e') = \lfloor log_c \frac{cost(e')}{pot(e')} \rfloor,$$

which implies that e' will not be considered as a candidate in the next d iterations of the *Greedy* algorithm of Section 5.2.6.1. This definition of delay, makes the (false) assumption that the information gain of an edge can only increase by a factor of $c > 1$ in each iteration, where the parameter c is used to control the penalty of having high sampling cost and having low information gain. As an example, assume an edge e' having an information gain of only 1% of the selected best edge e, and requiring to sample a new bi-connected component involving 10

edges upon probing. Also, let us assume that the information gain per iteration (and thus by insertion of other edges in the graph), may only increase by a factor of at most $c = 2$. We get $d(e') = \lfloor log_2 \frac{10}{0.01} \rfloor = \lfloor log_2 1000 \rfloor = 9$. Thus, using delayed sampling and having $c = 2$, edge e' would not be considered in the next nine iterations of the edge selection algorithm. It must be noted that this delayed sampling strategy is a heuristic only, and that no correct upper-bound c for the change in information gain can be given. Consequently, the delayed sampling heuristic may cause the edge having the highest information gain not to be selected, as it might still be suspended. The experiments show that even for low values of c (i.e., close to 1), where edges are suspended for a large number of iterations, the loss in information gain is fairly low.

5.2.7 Evaluation

In this section, the thesis empirically evaluates efficiency and effectiveness of the proposed solutions to compute a near-optimal subgraph of an uncertain graph to maximize the information flow to a source node Q, given a constrained number of edges, according to Definition 21. As motivated in the introducing Section 5.2.1, two main application fields of information propagation on uncertain graphs are: **i)** information/data propagation in spatial networks, such as wireless networks or a road networks, and **ii)** information/belief propagation in social networks. These two types of uncertain graphs have extremely different characteristics, which require separate evaluation. A spatial network follows a locality assumption, constraining the set of pairwise reachable nodes to a spatial distance. Thus, the average shortest path between a pair of two randomly selected nodes can be very large, depending on the spatial distance. In contrast, a social network has no locality assumption, thus, allowing to move through the network with very few hops. As a result, without any locality assumption, the set of nodes reachable in k-hops from a query node may grow exponentially large in the number of hops. In networks following a locality assumption, this number grows polynomial, usually quadratic (in sensor and road networks on the plane) in the range k, as the area covered by a circle is quadratic to its radius. The experiments have shown, that the locality assumption - which clearly exists in some applications - has tremendous impact on the performance of the algorithms proposed in this thesis, including the baseline. Consequently, both cases are evaluated separately.

All experiments were evaluated on a system with Linux 3.16.7, x86_64, Intel(R) Xeon(R) CPU E5,2609, 2.4GHz. All algorithms were implemented in Python 2.7. Dependencies: NetworkX 1.11, numpy 1.13.1.

5.2.7.1 Dataset Descriptions

This section describes the employed uncertain graph datasets. For both cases, i.e., with locality assumption and no-locality assumption, synthetic and real datasets were used.

Synthetic Datasets: No locality assumption. The first model *Erdös* is based on the idea of the *Erdös-Rényi model* [53], distributing edges independently and uniformly between nodes. Probabilities of edges are chosen uniformly in $[0,1]$ and weights of nodes are integers selected uniformly from $[0,10]$. It is known that this model is not able to capture real human social networks [132], due to the lack of modeling long tail distributions produced by "social animals". Thus, this data generation is only used in the first set of experiments and real social network data is employed later.

Synthetic Datasets: Locality assumption. Two synthetic data generating scheme to generate spatial networks are used. For the first data generating scheme - denoted by *partitioned* -, each vertex has the same degree d. The dataset is partitioned into $n = 2 \cdot \frac{|\mathcal{V}|}{d}$ partitions $P_0, ..., P_{n-1}$ of size d. Each vertex in partition P_i is connected to all and only vertices in the previous and next partition $P_{(i-1) \mod n}$ and $P_{(i+1) \mod n}$. This data generation allows to control the diameter of a resulting network, which is guaranteed to be equal to $n-1$.

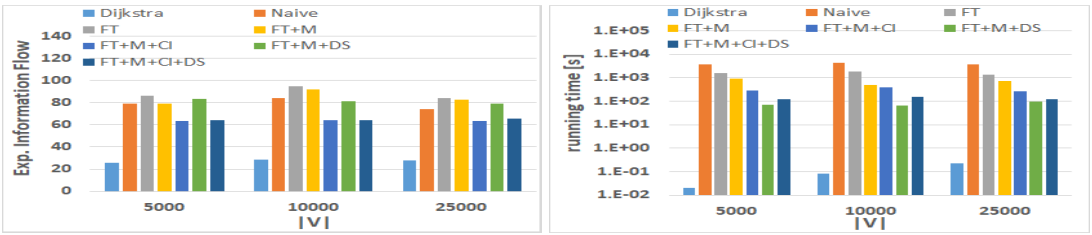
For a more realistic synthetic data set - denoted as *WSN* -, a wireless sensor network is simulated. Here, vertices have two spatial coordinates selected uniformly in $[0,1]$. Using a global parameter ε, any vertex v is connected to all vertices located in the ε-distance of v using Euclidean distance. For both settings, the probabilities of edges are chosen uniformly in [0, 1].

Real Datasets: No locality assumption. This thesis uses the *social circles of Facebook dataset* published in [131]. This dataset is a snapshot of the social network of Facebook - containing a subgroup of 535 users which form a social circle, i.e., a highly connected subgraph, having $10k$ edges. These users have excessive number of 'friends'. Yet, it has been discussed in [219] that the number of real friends that influence, affect and interact with an individual is limited. According to this result, and due to the lack of better knowledge which people of this social network are real friends, we apply higher edge probabilities uniformly selected in $[0.5;1.0]$ to 10 random adjacent nodes of each user. Due to symmetry, an average user has 20 such high probabilities 'close friends'. All other edges are assigned edge probabilities uniformly selected in $]0;0.5]$.

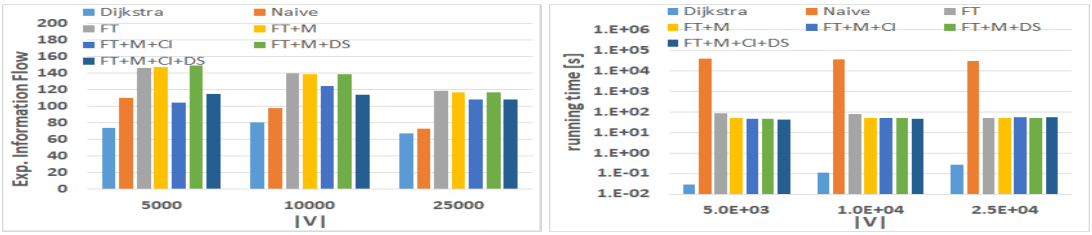
For the experiments on *collaboration network data*, the computer science bibliography *DBLP* is used. The structure of this dataset is such that if an author v_i co-authored a paper with author v_j, where $i \neq j$, the graph contains a undirected edge e from v_i to v_j. If a paper is co-authored by k authors this generates a completely connected (sub)graph (clique) on k nodes. This dataset has been published in [227]. Probabilities on edges are uniformly distributed in $[0;1]$. The graph consists of $|\mathcal{V}| = 317,080$ vertices and $|\mathcal{E}| = 1,049,866$ edges.

Finally, the proposed methods were also evaluated on the *YouTube social network*, first published in [146]. In this network, edges represent friendship relationships between users. The graph consists of $|\mathcal{V}| = 1,134,890$ vertices and $|\mathcal{E}| = 2,987,624$ edges. Again, the probabilities on edges are uniformly distributed in $[0;1]$.

Real Datasets: Locality assumption. For the experiments on *spatial networks*, the road net-



(a) Changing Graph Size with locality assumption



(b) Changing Graph Size without locality assumption

Figure 5.10: *Experiments with changing graph size*

work of San Joaquin County[23] was used, having $|\mathscr{V}| = 18,263$ vertices and $|\mathscr{E}| = 23,874$ edges. The vertices of the graph are road intersections and edges correspond to connections between them. In order to simulate real sensor nodes located at road intersections, we have connected vertices that are spatially distant from each other have a lower chance to successfully communicate. To give an example, for two vertices having a distance of a in meters, the communication probability is set to $e^{-0.001a}$. Thus, a $10m$, $100m$ and $1km$ distance will yield a probability of $e^{-0.01} = 99\%$, $e^{-0.1} = 90\%$, and $e^{-1} = 36\%$, respectively.

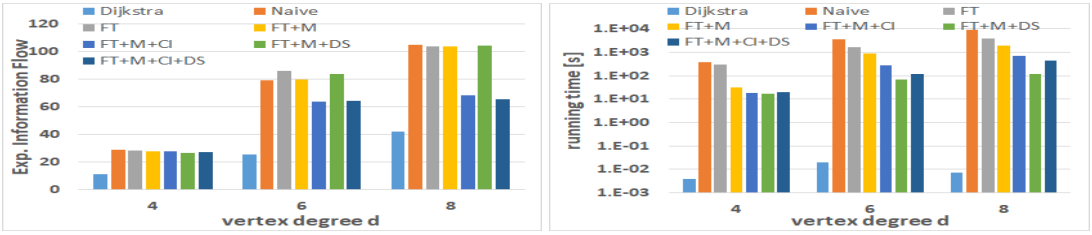
5.2.7.2 Evaluated Algorithms

The algorithms that were evaluated in this section are denoted and described as follows:

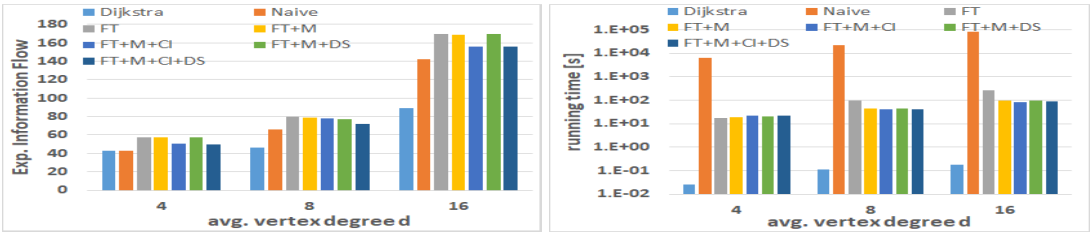
Naive As proposed in [130, 52] the first competitor *Naive* does not utilize the strategy of component decomposition of Section 5.2.5 and utilizes a pure sampling approach to estimate reachability probabilities. To select edges, the *Naive* approach chooses the locally best edge, as shown in Section 5.2.6, but does not use the *F-tree* representation presented in Section 5.2.5.3. A constant Monte-Carlo sampling size of 1000 samples is used.

Dijkstra Shortest-path spanning trees [193] are used to interconnect a wireless sensor network (WSN) to a sink node. To obtain a maximum probability spanning tree, we proceed as follows: the cost $w(e)$ of each edge $e \in \mathscr{E}$ is set to $w(e) = -\log(P(e))$. Running the traditional Dijkstra algorithm on the transformed graph starting at node Q yields, in each iteration, a spanning tree which maximizes the connectivity probability between Q and any node connected to Q [187]. Since, in each iteration, the resulting graph has a tree structure, this approach can fully exploit the concept of Section 5.2.5, requiring no sampling step at all.

[23] https://www.cs.utah.edu/ lifeifei/SpatialDataset.htm



(a) Changing Graph Density with locality assumption



(b) Changing Graph Density without locality assumption

Figure 5.11: *Experiments with changing graph density*

FT employs the *F-tree* proposed in Section 5.2.5.3 to estimate reachability probabilities. To sample bi-connected components, 1000 samples were drawn for a fair comparison to *Naive*. All following *FT*-Algorithms build on top of *FT*.

FT+M additionally memoizes for each candidate edge e the pdf of the corresponding bi-connected component from the last iteration (cf. Section 5.2.6.2).

FT+M+CI further ensures that probing of an edge is stopped whenever another edge has a higher information flow with a certain degree of confidence, as explained in Section 5.2.6.3.

FT+M+DS instead tries to minimize the candidate edges in an iteration by leaving out edges that had a small information gain/cost-ratio in the last iteration (cf. Section 5.2.6.4). Per default, the penalization parameter is set to $c = 2$.

FT+M+CI+DS is a combination of all the above concepts.

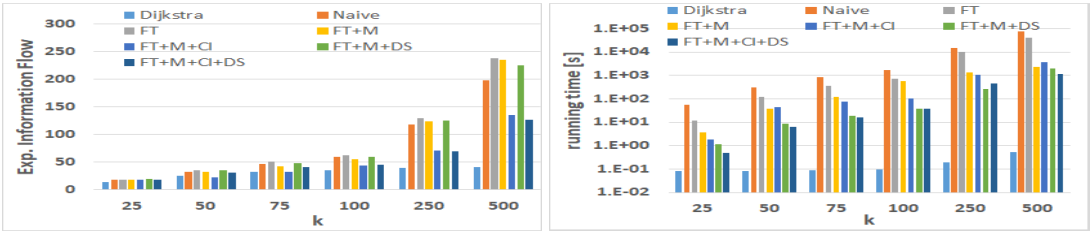
5.2.7.3 Experiments on Synthetic Data

This section employs randomly generated uncertain graphs. We generate graphs having no-locality-assumption using *Erdös* graphs and having locality assumption using the *partitioned* generation. Both generation approaches are described in Section 5.2.7.1. This data generation allows us to scale the topology of the uncertain graph $\mathcal{G}$ in terms of size and density. Unless specified otherwise, a graph size of $|\mathcal{V}| = 10,000$ is used, a vertex degree of 6 and a budget of edges $k = 200$ in all experiments on synthetic data.

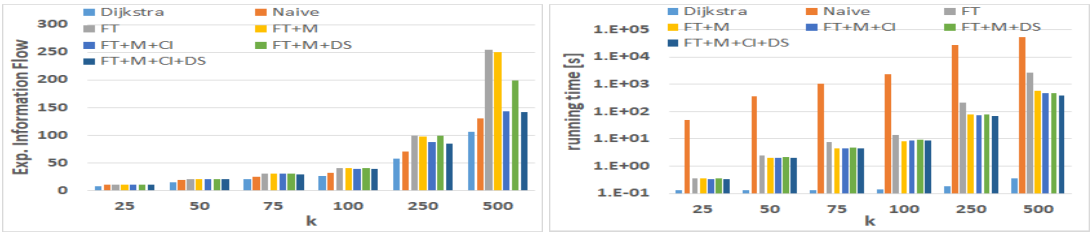
Graph Size. We first scale the size $|\mathcal{V}|$ of the synthetic graphs. Figure 5.10a shows the information flow (left-hand-side) and running time (right-hand-side) for the synthetic data set following the locality assumption. First, we note that the *Dijkstra*-based shortest-path spanning tree yields an extremely low information flow, far inferior to all other approaches. The reason

is that a spanning tree allows no room for failure of edges: whenever any edge fails, the whole subtree become disconnected from Q. We further note that all other algorithms, including the *Naive* one, are oblivious to the size of the network, in terms of information flow and running time. The reason is that, due to the locality assumption, only a local neighborhood of vertices and edges is relevant, regardless of the global size of the graph. Additionally, we see that the delayed sampling heuristic (*DS*) yields a significant running time performance gain, whilst keeping the information flow constantly high. The combination of all heuristics (*FT+M+CI+DS*) yields significant loss of information flow due to the pruning strategy of the confidence interval heuristic (*CI*). Figure 5.10b, shows the performance in terms of information gain and running time for the *Erdös* graphs having no locality assumption. The first observation is that *Dijkstra* and *Naive* yield a significantly lower information flow than the proposed approaches. For *Dijkstra*, this result is again contributed to the constraint of constructing a spanning tree and, thus, not allowing any edges to connect the flow between branches. For the *Naive* approach, the loss in information flow requires a closer look. This approach samples the whole graph only 1000 times, to estimate the information flow. In contrast, the *F-tree* approach samples each individual bi-connected component 1000 times. *Why is the later approach more accurate?* A first, informal, explanation is that, for a constant sampling size, the information flow of a small component can be estimated more accurately than for a large component. Intuitively, sampling two independent components n times each, yields a total of n^2 combinations of samples of their joint distribution. More formally, this effect is contributed to the fact that the variance of the sum of two random variables increases as their correlation increases, since $Var(\sum_{i=1}^{n} X_i) = \sum_{i=1}^{n} Var(X_i) + 2\sum_{1 \leq i < j \leq n} Cov(X_i, X_j)$ [150]. Furthermore, the *Naive* approach also incurs an approximation error for mono-connected components, for which all *F-tree* (*FT*) approaches compute the exact flow analytically. We further see that the *Naive* approach, which has to sample the whole graph, is by far the most inefficient. On the other end of the scope, the *Dijkstra* approach, which is able to avoid sampling entirely by guaranteeing a single mono-connected component, is the fastest in all experiments, but at the cost of information flow. We also see that in Figure 5.10b all algorithms stay in the same order of magnitude in their running time and information flow as the graph increases. This is due to the fact that in this experiment we stay constant in the average vertex degree, i.e., $deg(v_i) \approx 10$ for all $v_i \in \mathcal{V}$. We also observe that the *CI* heuristic yields an overhead of computing the intervals whilst losing information gain due to its rigorous pruning strategy.

Graph Density. In this experiment, the average degree of vertices is scaled. In the case of graphs following the locality assumption, the gain in information flow of all proposed solutions compared to *Dijkstra* is quite significant as shown in Figure 5.11a, particularly when the degree of vertices is low. This is the case in road networks, but also in most sensor and ad hoc networks. The reason is that, in such case, spanning trees gain quickly in height as edges are added, thus, incurring low-probability paths that require circular components to connect branches to support the information flow. For larger vertex degrees, the optimal solutions become more star-like, thus, becoming more tree-like. For a small vertex degree, we observe also that the same bi-



(a) Changing budget k with locality assumption

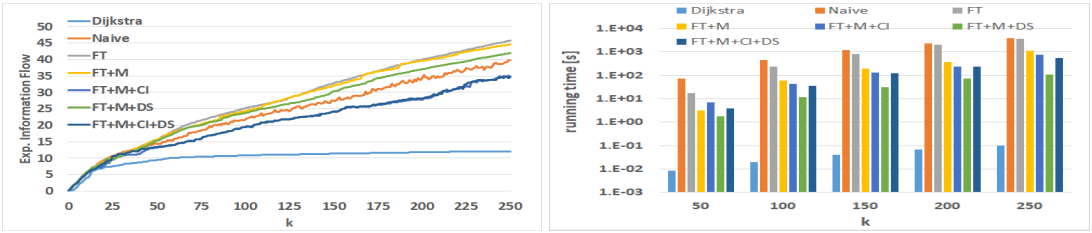


(b) Changing budget k without locality assumption

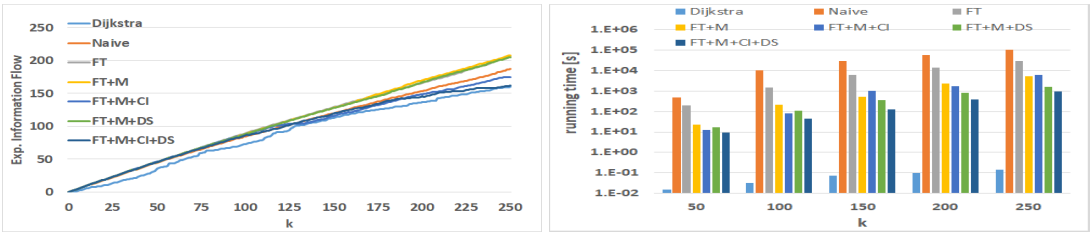
Figure 5.12: *Experiments with changing Budget*

connected-components are occurring in consecutive iterations resulting in a running time gain for the memoization approach. As the complexity of the graph grows, the gap between the *FT* and *FT+M* shrinks as more candidates result in an increased number of possibilities where bi-connected components can occur and, thus, make cached results for bi-connected components obsolete. The results shown in Figure 5.11b indicate that the *Dijkstra* approach is able to find higher quality result in graphs without locality assumption for small (average) vertex degrees. This is contributed to the fact that in graphs with this setting, the optimal result will be almost tree-like, having only a few inter-branch edges. The algorithms *FT+M+CI* and *FT+M+CI+DS* yield a trade-off between running time and accuracy, i.e., we observe a slightly loss in information gain coming along with a better running time for a setting with a larger (average) vertex degree.

Scaling of parameter k**.** In the next experiments, shown in Figure 5.12, the thesis shows how the budget k of edges affects the performance of the proposed algorithms. In the case of a network following the locality assumption, we observe in Figure 5.12a that the overall information gain per additional edge slowly decreases. This is clear, since in average, the hop distance to Q increases as more edges are added, increasing the possible links of failure, thus, decreasing the likelihood of propagating a nodes information to Q. We observe that the effectiveness of *Dijkstra* in the locality setting quickly deteriorates, since the constraint of returning a tree structure leads to paths between Q and other connected nodes that become increasingly unlikely, making the *Dijkstra* approach unable to handle such settings. Here, the memoization heuristic *M* performs extremely well. A sever loss of information gain is observed when running *FT+M+CI* and *FT+M+CI+DS* due to its pruning policy. Later one is the best in terms of running time.



(a) WSN $\varepsilon = 0.05$



(b) WSN $\varepsilon = 0.07$

Figure 5.13: *Experiments in synthetic Wireless Sensor Networks*

In contrast, using a network following no locality assumption in Figure 5.12b, we see that both *Dijkstra* and *Naive* yield an low information gain for a large budget k. For *Dijkstra*, the reason is that for large values of k, the depths of the spanning tree, which is lower bounded by $log_d k$, incurs longer paths without any backup route in the case of a single failure. For the *Naive* approach, the low information gain is contributed to the high variance of sampling the information flow of the whole graph for each edge selection. Further, we see that the *Naive* approach further suffers from an extreme running time, requiring to re-sample the whole graph in each iteration. The F-tree in combination with the memoization give a consistently high information gain while having a low running time. The heuristics suffering from a loss in information gain yield a slightly better running time.

Synthetic Wireless Sensor Networks (WSN). In these experiments, real world wireless sensor networks (WSN) are simulated. We embed a number of vertices - here $|\mathcal{V}| = 1,000$ - according to a uniform distribution in a spatial space $[0;1] \times [0;1]$. For each vertex, we observe adjacent vertices being in its proximity which is regulated by an additional parameter ε. Figure 5.13 shows the results. We observe nearly the same behavior as in Figure 5.11a. As the parameter ε is a regulator for the graph's interconnectivity, we observe again a fair trade-off of information gain and running time for the proposed heuristics. By increasing the parameter ε, hence, simulating dense graphs, the gap between *Dijkstra* and the *F-tree* approaches is reduced. For these datasets, we can also observe the benefit of *FT+M+CI+DS* which still identifies a high information gain whilst reducing the running time, as the number of candidates are reduced, respectively, we can prune candidates in earlier stages of each iteration.

(a) San Joaquin County Road Network

(b) Circle of Friends - Facebook

(c) DBLP

(d) YouTube

Figure 5.14: *Experiments on Real World Datasets*

Parameter c**.** The thesis also evaluated the penalization parameter c of the delayed sampling heuristics and summarizes the results. In all evaluated settings, ranging from $1.01 \geq c \geq 16$, the running time consistently decreases as c is decreased, yielding a factor of 2 to 10 speed-up for $c = 1.2$, depending on the dataset, and a multi-orders of magnitude speed-up for $c = 1.01$. Yet, for $c < 1.2$ we start to observe a significant loss of information flow. For the extreme case of $c = 1.01$, the information flow became worse than *Dijkstra*, as edges become suspended unreasonable long, choosing edges nearly arbitrarily. For the default setting of $c = 2$ used in all previous evaluations, the *delayed sampling heuristics* showed insignificant loss of information, but yielding a better running time.

5.2.7.4 Experiments on Real World Data

The first real world data experiments use the *San Joaquin County Road Network*. As road networks are of very sparse nature, and follow a strong locality assumption, the proposed approaches outperforms *Dijkstra* significantly as k is scaled to $k = 250$. Thus, *Dijkstra* is highly undesirable as budget is wasted without proper return in information flow. In this setting, following the locality assumption, we see that all heuristics yield a significant run time performance gain, while the information flow remains similar for all heuristics.

In the next experiment, the *social circles of friends* dataset is employed, an extremely dense network with no locality assumption, where most pairs of nodes are connected. As described in Section 5.2.7.1, each vertex in this graph only has ten high-probability links, whereas all other nodes have a lower probability. Figure 5.14b shows that *Dijkstra* yields a most significant loss of information, as it is forced to quickly build a large-height tree to maintain high probability edges. Further, we see that the memoization heuristic yields a significant running time improvement of about one order of magnitude. Notably, in such dense settings, heuristics *CI* and *DS* show almost no effect in both, runtime and information flow.

Figure 5.14c shows similar results on the *DBLP* collaboration network dataset, a sparse network which follows no locality assumption. Again, we observe a loss of potential information flow for *Dijkstra* as k increases.

Finally, we observe similar behavior of all approaches on a bigger graph such as the *YouTube social network*, which refers to a sparse setting with no locality assumption. Figure 5.14d shows the results. As in the other settings, we can observe an extremely low information flow of *Dijkstra*, and an extreme running time of the *Naive* approach. It is interesting that in this setting, the memoization approach *FT+M* yields only a minimal gain in running time, like the other heuristics. Fortunately, none of these heuristics shows a significant loss of information flow.

5.2.7.5 Experimental Evaluation: Summary

To summarize the experimental results of the thesis, let us reiterate the shortcomings of the naive solutions, and briefly discuss which of the heuristics are best used in which setting.

Naive: The *Naive* competitor, which applies a Greedy edge selection (c.f. Section 5.2.6) but does not use the *F-tree*, is multiple orders of magnitude slower than other approaches in all real-data experiments (cf. Figure 5.14). Further, large sampling errors also yield a significantly lower information flow in most settings.

Dijkstra: A *Dijkstra*-based spanning tree algorithm runs extremely fast, but at the cost of an extreme loss of information, yielding low information flow. The information loss is particularly high for social networks (e.g., Figure 5.14b), where cycles are required to increase the odds of connecting a distant node to the source.

FT: Employing the *F-tree* proposed in Section 5.2.5.3 maximizes the information flow. Compared to the *Naive* approach, smaller partitions need to be sampled yielding smaller sampling variation while being multiple orders of magnitude faster.

FT+M: The memoization heuristic technique described in Section 5.2.6.2 was shown to be simple and effective. It yields vast reduction in running time of up to one order of magnitude on real-data (see Figure 5.14), while showing no notable detriment to the information flow.

FT+M+CI: Employing confidence intervals as described in Section 5.2.6.3 has shown a significant improvement in running time on spatial networks following the locality assumption (cf. Figures 5.10a, 5.11a, and 5.14a). However, this heuristic yields no improvement (and often has a detrimental effect) in settings without locality assumptions such as in social networks (cf. Figure 5.14b-5.14d). This heuristic should not be employed in such settings.

FT+M+DS: The delayed sampling heuristic presented in Section 5.2.6.4 yields an improvement in running time in networks following the locality assumption. This improvement is especially large in cases having a high vertex degree (cf. Figure 5.11a). However, in social networks which do not follow the locality assumption, the gain of this heuristic is often marginal (cf. Figure 5.14b-5.14d). Yet, this heuristic comes at minimal loss of information flow, such that it is not detrimental to enable it by default.

FT+M+CI+DS: The combination of all heuristics inherits the problems of FT+M+CI and FT+M+DS for the cases without locality assumption. But for the cases with locality assumption, the experiments on real world data show that in most cases the combination of all heuristic achieves significant lower running time compared to setting where each of the heuristics is applied separately proofing the importance of each proposed heuristic.

5.3 Summary

In this chapter, the thesis discussed solutions for the problem of maximizing the information flow in an uncertain graph given a fixed budget of k communication edges. It identified two *NP*-hard subproblems that needed heuristical solutions: **(i)** Computing the expected information flow of a given subgraph; and **(ii)** selecting the optimal k-set of edges. For problem *(i)*, an advanced sampling strategy was developed that only performs an expensive (and approximative) sampling step for parts of the graph for which we can not obtain an efficient (and exact) analytic solution. For problem *(ii)*, the thesis proposes the *F-tree* representation of a graph $\mathcal{G}$, which keeps track of *bi-connected components* - for which sampling is required to estimate the information flow - and *mono-connected components* - for which the information flow can be computed analytically. On the basis of the *F-tree* representation, further approaches and heuristics were introduced to handle the trade-off between effectiveness and efficiency. The evaluation shows that these enhanced algorithms are able to find high quality solutions (i.e., k-sets of edges having a high information flow) in efficient time, especially in graphs following a locality assumption, such as road networks and wireless sensor networks.

6

Spatial Graphs

If people do not believe that mathematics is simple,
it is only because they do not realize how complicated life is.

John von Neumann
1903-1957

Attribution

This Chapter uses material from the following publication:

- Christian M.M. Frey, Alexander Jungwirth, Markus Frey, and Rainer Kolisch. The vehicle routing problem with time windows and flexible delivery locations. *European Journal of Operational Research*, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.11.029. URL `https://www.sciencedirect.com/science/article/pii/S0377221722008888`. [66]

See § 1.4 for an overview of incorporated publications and the author's attribution.

Highlights

- New type of vehicle routing with multiple capacitated service locations;
- Hybrid self-adapting metaheuristic quickly reaching good regions of the solution space;
- Backtracking mechanism to revert unsatisfactory decisions early in construction phase;
- New neighborhoods exploiting flexibility of service locations;
- Algorithm is effective and efficient on realistic hospital data;

6.1 Operational Research for Routing Problems

The *Traveling Salesman Problem* (TSP) is known to be one of the most classical optimization problems. The challenge is to compute an optimal sequence of nodes that minimizes the costs of visiting all nodes that define a path in a graph, starting from a default source node and possibly returning to this node. The extension of this fundamental problem is known under the terminology of *Vehicle Routing Problems* (VRPs) [202, 203, 125], where the task is to visit these nodes using one or several (heterogeneous) fleets of vehicles. From a practical point of view, these nodes can describe depots, cities, customers' locations, and so on. The problem can be seen as a combination of two subtasks. First, which nodes have to be allocated to a vehicle defining a path (route), and second, what is the optimal solution of the TSP over this subset of nodes? Depending on the problem setting, VRPs come in different complexity levels. For example, we can define resource constraints on the vehicle-specific traveling times from one node to another or on the capacities associated with the carrying load. One can also define time-window constraints for time intervals where a pickup and delivery service must be scheduled or if a vehicle can serve only one or several routes.

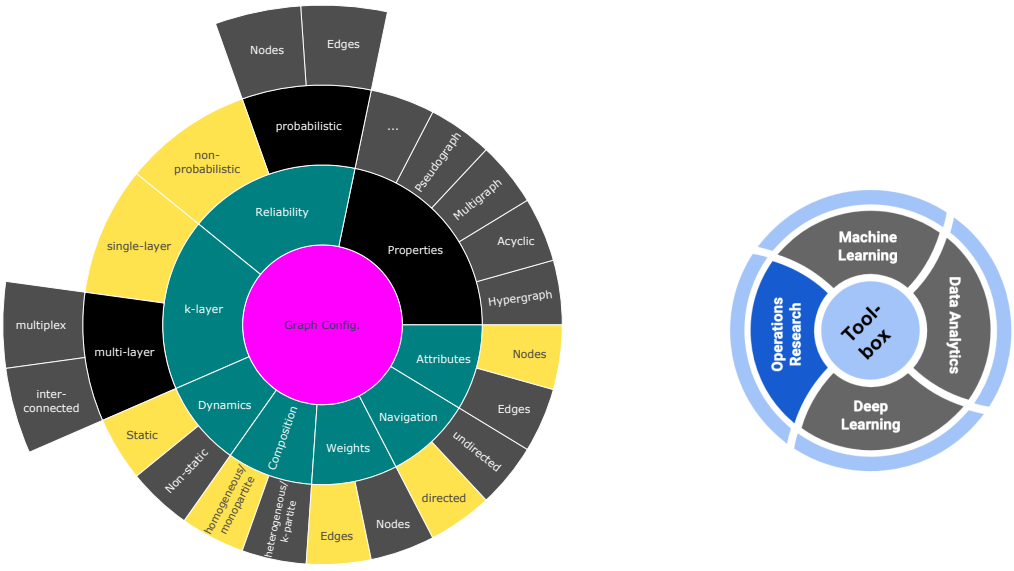
The following chapter is devoted entirely to solving an optimization problem under the lens of operational research using graph-theoretic tools. It focuses on a vehicle routing problem where several constraints are defined. The problem is extended by time-window constraints stating that customers have to be served within a specific time interval. This is commonly known as *Vehicle Routing Problem with Time Windows* (VRPTW). Furthermore, the problem is extended by flexible locations, meaning that customers gain the flexibility that they can be served in several capacitated locations. The practical relevance arises, for example, in delivery services, where customers can receive their sendings in several locations, e.g., at home, at a pickup station, or working place. Hence, the thesis presents a novel heuristic to solve the vehicle routing problem with time windows in a spatial network with flexible locations (VRPTW-FL) of customers served by a set of heterogeneous vehicles.

6.2 Metaheuristic for VRPTW-FL

The thesis discusses a new variant of the well-known vehicle routing problem (VRP): the VRP with time windows and flexible delivery locations (VRPTW-FL). Generally, in the VRP each customer is served in one fixed service location. However, in the VRPTW-FL each customer is served in one of a set of potential service locations, each of which has a certain capacity. From a practical point of view, the VRPTW-FL is highly relevant due to its numerous applications, e.g., parcel delivery, routing with limited parking space, and hospital-wide scheduling of physical therapists. Theoretically, the VRPTW-FL is challenging to solve due to the limited location capacities. When serving a customer, location availability must be ensured at every time.

To solve this problem, the thesis presents a mathematical model and a tailored hybrid adaptive large neighborhood search. The proposed heuristic makes use of an innovative backtracking approach during the construction phase to alter unsatisfactory decisions at an early stage. In the meta-heuristic phase, novel neighborhoods are employed and updates of the objective violation weights are dynamically adapted. For the computational analysis, hospital data is used to evaluate the utility of flexible delivery locations and various cost functions. The proposed algorithmic features improve the solution quality considerably.

Graph Configuration.
The configuration used to solve the routing problem being proposed in this chapter, consists of a single-layer, non-probabilistic, static, monopartite graph as illustrated in Figure 6.1a. The routing of vehicles through the network are expressed via directed, weighted edges where weights denote the distance between two nodes, respectively, the locations within the routing network. The proposed problem is discussed in the realm of capacitated locations, hence, each node contains attributive information about its maximal capacity. To solve the scheduling problem, a metaheuristic is proposed which is affiliated with the technical domain of *Operations Research* (OR).



(a) *Graph Configurator* ***(b)*** *Toolbox*

Figure 6.1: *Graph Configuration and methodology's classification used to solve the vehicle routing problem with time windows and flexible locations (VRPTW-FL)*

6.2.1 Introduction

Vehicle routing is well-studied in the operations research and management science literature. It has theoretical as well as practical relevance to scientific communities and industries, such as logistics and healthcare. In the classic *vehicle routing problem* (VRP), vehicles traverse a network with the objective to, e.g., minimize routing costs or the number of vehicles used. Each destination in the network corresponds to exactly one customer, and each customer is visited once. For the VRP, several extensions exist on the demand and delivery side. For example, on the delivery side assigning capacities to the vehicles leads to the *capacitated VRP*, while on the demand side associating customers with time windows leads to the *VRP with time windows* (VRPTW) [38].

In this chapter, the thesis presents an extension of the VRP with substantial enhancement of the demand side: the *VRP with flexible delivery locations* (VRP-FL). In this problem, a customer is no longer automatically assigned to his/her service location. Instead, in the VRP-FL each customer must be served at exactly one *capacitated* location among a set of multiple alternatives. In this context, capacitated means that the number of customers, which can be served at one location at the same time is limited. When, additionally, time windows for customers are considered, we obtain the *VRP with time windows and flexible delivery locations* (VRPTW-FL). Note that by assigning capacities to service locations, the complexity of the problem increases significantly. Non-availability of locations leads to rerouting of customers to alternative service location. Thus, the location capacity directly influences the routing decision.

There is little literature on VRPs incorporating flexible customer locations. This thesis presents a pioneering study of this type of problem with capacitated locations from a theoretical side and introduces a compact mathematical definition of the problem. The VRPTW-FL has been inspired by a problem in the health care industry, where it is known as the hospital-wide therapist scheduling and routing problem [70]. Hospital planners have to decide which therapist treats which patient in which room at which time. Therapists can treat patients either at the ward or in a therapy center. For the VRPTW-FL, vehicles represent therapists, customers represent patients and locations for the customers represent treatments rooms. Especially in larger hospitals, the travel times of therapists are considerable. Reducing travel times allows more time to treat patients, which in the long run reduces the average waiting time for an appointment.

Another application of the VRPTW-FL is flexible parcel delivery. Companies such as DHL and Amazon have experimented with delivering to different locations depending on the time of the day [8]. For example, a parcel can be sent to a customer's home, the trunk of the customer's car or to a parcel box.

This thesis presents a *mixed integer program* (MIP) for the VRPTW-FL. As a generalization of the VRP, the VRPTW-FL is also *NP*-hard. It is shown that the VRPTW-FL can be described with a limited number of linear constraints when location capacities are modeled as resource flows. However, this problem is extremely hard to solve to optimality. Therefore, to tackle

the problem a *hybrid meta-heuristic* based on *adaptive large neighborhood search* (ALNS) and *guided local search* (GLS) is employed.

The construction heuristic is based on insertion, and a backtracking mechanism is added to alter unsatisfactory decisions at an early stage. The solution derived by the construction heuristic is then further improved by the hybrid ALNS. The self-adaptiveness of the ALNS is extended by allowing feasibility violations which are penalized in the objective function. The penalty weights are dynamically adjusted following a GLS approach, which adds robustness to the ALNS, and may make it more suitable for future applications.

In the computational study, the algorithm is assessed from a theoretical and a practical perspective. In the theoretical part, the performance of the heuristic procedure is examined in general and its new features in particular. In the practical part, the heuristic is tested against current hospital planning, and the potential benefit of flexibility is evaluated by applying different cost functions for serving customers in different locations and are put into relation to the vehicles' travel costs.

The results show that the heuristic works well; combining the ALNS with a GLS leads the heuristic to considerably better regions of the planning horizon, and backtracking provides much better initial solutions than traditional construction heuristics. When applied to the hospital case, the heuristic clearly outperforms current hospital planning methods. In general, the results encourage planners facing similar problems to consider some degree of location flexibility whenever possible.

Outline. The remainder of this chapter is structured as follows. It begins in § 6.2.2 with an overview of related work focusing on VRPs incorporating location decisions. In § 6.2.3, a mathematical formulation for the VRPTW-FL is developed and the underlying graph structure is discussed. In § 6.2.4, the hybrid meta-heuristic procedure is presented which is used to solve the problem. Evidence of the algorithm's capabilities is provided in § 6.2.5 and the chapter concludes in § 6.3.

6.2.2 Related work

The VRPTW-FL is inspired by the therapist scheduling and routing problem (ThSRP) presented in Gartner et al. [70]. Gartner et al. [70] put the focus on the scheduling perspective and model the problem as a variant of the resource constraint project scheduling problem, see, e.g., Brucker et al. [23]. The problem is solved to optimality by adding routing cuts to a scheduling problem being called each time a cut is introduced. Service and travel times are multiples of 30 minute time buckets. More granular intervals, e.g., five minutes, would lead to intractable computing times as the number of variables in the scheduling problem would exponentially grow. In contrast, by modeling the ThSRP as a VRP instead of a scheduling problem, it is possible to use existing efficient solution approaches already successfully applied to rich VRPs, like the ALNS (see [161]),

allowing more granular time buckets and larger instances. Jungwirth et al. [103] work on the same definition of the VPRTW-FL as this thesis. The authors focus firmly on the application of hospital therapist scheduling and develop a computational demanding branch-price-and-cut algorithm. In contrast to Jungwirth et al. [103], the thesis studies the VPRTW-FL also from a theoretical side, i.e., it presents a compact mathematical definition of the problem, and, moreover, it highlights the structural differences with the classic VRPTW and derives implications on optimality bounds.

The VRP and its extensions have been studied extensively in the literature. Textbooks include Toth and Vigo [202, 203] as well as Golden et al. [76], while literature reviews are provided by, e.g., Desrochers et al. [39], Laporte and Osman [124], Desrochers et al. [40], Cordeau et al. [34], Eksioglu et al. [51], Laporte [123], Lahyani et al. [120] and Vidal et al. [210]. The review presented here focuses on routing problems incorporating location decisions being the major feature of the VRPTW-FL.

The first works considering both location and routing aspects date back to the 1960s, e.g., Maranzana [144], von Boventer [211], Webb [217], Watson-Gandy and Dohrn [216]. Since then a multitude of different problems have arisen, all having routing and location decisions (see, e.g., [165]). However, to the best of the author's knowledge, this thesis is the first presenting a mathematical formulation for the problem encompassing time-dependent capacitated and flexible locations for customers. In literature, we find four problem types in the field of VRPs showing similarities to the presented problem: (1) the *vehicle routing-allocation problem* (VRAP) introduced by Beasley and Nascimento [15], (2) the *generalized VRP* (GVRP) introduced by Ghiani and Improta [71], (3) the *VRP with synchronization constraints* (VRPSC), see Drexl [44] for a general overview and taxonomy of synchronization constraints, and (4) the electric VRP (E-VRP) or Green VRP discussed by, e.g., Froger et al. [67] and Bruglieri et al. [24], respectively.

(1) VRAP In the VRAP, being a special case of the location routing problem (LRP), no location capacities and time windows are considered. This is a significant difference to the problem being discussed in the following.

(2) GVRP The GVRP is an extension of the VRP in which vehicles visit clusters of potential delivery sites instead of individual customers. Each cluster has a given demand and only one delivery site in the cluster must be visited, e.g., when routing vessels in maritime transportation, only one port in a certain region is visited; see, e.g., Baldacci et al. [12], Bektaş et al. [17]. In contrast to the VRPTW-FL, the locations of customers are disjoint from each other, the customers' time windows span across the entire planning horizon and multiple visits of a locations are not possible. Moccia et al. [147] introduce the *GVRP with time windows* (GVRPTW), where an individual time window is assigned to each node within a cluster, i.e., the time windows are location and not customer specific. In the *VRP with roaming delivery locations*, a special case of the GVRPTW (see [167, 154]), multiple service locations can exist for a customer. However, the

location capacities are set to one and the time windows for the nodes in one cluster are disjoint from each other. The VRP with delivery options represents an extension of the GVRPTW by incorporating non-renewable location capacities (see [47, 48, 200]).

(3) VRPSC Using the taxonomy of Drexl [44], the time-dependent location capacities of the VRPTW-FL belong to resource synchronization while the precedence constraints belong to operations synchronization. With synchronization constraints, tours are not independent anymore, i.e., a change of one tour may affect other tours.

According to Drexl [44] resource synchronization is given if "*the total utilization or consumption of a specified resource by all vehicles must be less than or equal to a specified limit*" at any point in time. For example, the VRP of Lam and Hentenryck [121] considers location congestion where vehicles are synchronized with regard to, e.g., parking lots and forklifts. Operations synchronization are equivalent to precedence constraints (see [44]) occurring in many applications, e.g., in dial-a-ride problems and in pickup-and-delivery problems (see, e.g., [13, 42, 87]). However, there is no VRPSC incorporating also flexibility for customers' service locations.

(4) Green-VRP In the Green-VRP or E-VRP vehicles can be recharged between two customer visits at a number of capacitated charging stations; see Froger et al. [67] for E-VRPs, and Bruglieri et al. [24] for the Green-VRPs. However, these flexible charging locations are different from the service locations which are flexible in the problem being discussed. Charging stations could not be visited at all and the charging duration is a decision variable. Moreover, time windows, precedence relations and a heterogeneous fleet are not considered in the E-VRP.

6.2.3 Model Development

In this section, the mathematical model is developed and the underlying graph structure is discussed. Section 6.2.3.1 gives a formal problem description and introduces the notation. The thesis follows the standard notation for VRPs and VRPTWs as presented in [97] and [38], respectively. However, it deviates from their notation when necessary to model the special properties of the VRPTW-FL. In Section 6.2.3.2, the graph structure of the VRPTW-FL is detailed and its differences from the graph of the classic VRPTW is demonstrated. Finally, Section 6.2.3.3 presents a linear mixed integer problem formulation for the VRPTW-FL.

6.2.3.1 Formal Problem Description

The classic VRP serves a set of customers $\mathscr{I} = \{1, 2 \ldots, I\}$ with specific demand $q_i > 0$ for a single good using a set of homogeneous vehicles $\mathscr{K} = \{1, \ldots, K\}$ with given capacity $Q > 0$. A vehicle starts its tour in the depot, visits a subset of customers $\mathscr{S} \subseteq \mathscr{I}$ and returns to the depot, which is denoted as dummy customer 0 for the outward trip, and $I+1$ for the return trip. In both cases the demand is assumed to be $q_0 = q_{I+1} = 0$.

The connections between two customers i and j, including the depot as dummy customers, are associated with travel cost $c^{\text{travel}}_{l_i,l_j}$ with l_i and l_j being the service locations for customer i and j. The aggregated demand of the customers visited by a single vehicle must be less than or equal to the vehicle's capacity. The objective is to minimize the total travel costs over all vehicles while serving all customers.

The VRPTW extends the VRP by assigning a specific service time s_i and time window $[a_i, b_i]$ to each customer i, with a_i and b_i being the earliest and latest possible start of service, respectively. The travel time between two customers is denoted as $t_{i,j} \geq 0$. Generally, a hard time window restriction is used, which means a vehicle can arrive at the customer before a_i but never after b_i. In case of early arrival, the vehicle must wait at the site of customer i until a_i.

The VRPTW-FL extends the VRPTW by allowing additional locations for serving the customers. The set of locations is defined as $\mathscr{L} = \{0, \ldots, L\}$ and a customer i can be served in a subset of locations $\mathscr{L}_i \subseteq \mathscr{L} \backslash \{0\}$ with location 0 being the depot. Location $l \in \mathscr{L}$ has a capacity C_l defining the maximum number of customers which can be served at the same time. For unbounded locations, let $C_l = \infty$. When serving customer i at location l, fixed location costs $c^{\text{location}}_{i,l} \geq 0$ are incurred. In the hospital setting, location costs are 0 if the patient is treated at the ward and equal the cost for transporting the patient to the central treatment room, if the patient is treated there. The objective of the VRPTW-FL is to minimize the sum of travel and location costs, where travel costs $c^{\text{travel}}_{l,r}$ are defined as the cost of traveling between two locations $l, r \in \mathscr{L}$ instead of traveling between two customers $i, j \in \mathscr{I} \cup \{0\}$.

The crucial information in the VRPTW-FL is if a certain location $l \in \mathscr{L}$ is available for any arbitrary small time interval $\tau \in [a_\tau, b_\tau]$ with $0 \leq a_\tau \leq b_\tau$ or if the location capacity is already fully used. Therefore, an indicator function $I(i,l,k,\tau)$ is introduced which is 1 if customer i is served in location l by vehicle k in time interval τ and 0 otherwise. Using this indicator, the number of customers being served at a specific location in any given time interval is calculated.

Therapist scheduling as a practical application of the VRPTW-FL incorporates two additional aspects: *precedence relations* between customers and *heterogeneous vehicles*. Certain customers have to be visited before other customers can be visited. Note, in therapist scheduling a "customer" corresponds to a treatment and multiple treatments might be required for a single patient during the planning horizon. Some treatments have to be executed before other treatments can start, e.g., a cast must be removed before a stretching or strengthening exercise can be done. Therefore, a set $\mathscr{P}$ is defined as the precedence relations between customer tuple $\langle i, j \rangle$, in which customer i must be served before customer j can be served.

Therapists also differ in skills and shift patterns. Each therapist belongs to one of two shift types: regular shifts or short shifts. Furthermore, each therapist has a certain skill set which defines treatments that can be carried out by the therapist. Thus, it might be that a therapist is not qualified for a particular treatment or the shift pattern does not allow for visiting a customer during his/her time window. Therefore, therapists are modeled by heterogeneous vehicles and each vehicle k can service a subset of customers $\mathscr{I}_k \subseteq \mathscr{I}$.

6.2.3.2 Structural Differences of VRPTW and VRPTW-FL

Having introduced the basic notation, this section examines the structural differences between the VRPTW and the VRPTW-FL. A graphical representation for the VRPTW-FL is derived and it is shown that the optimal objective function value of the VRPTW always yields an upper bound for the VRPTW-FL.

To represent the VRPTW-FL as a network, a directed graph $G = (\mathscr{V}, \mathscr{A})$ with vertex set $\mathscr{V}$ and arc (=edge; cf § 2.1) set $\mathscr{A}$ is used. In the proposed problem, each vertex corresponds to a customer-location tuple $\langle i,l\rangle \in \{\mathscr{I} \cup \{0\}\} \times \mathscr{L}_i$. For arc set $\mathscr{A} \subseteq \mathscr{V} \times \mathscr{V}$, we have $\langle\langle i,l\rangle, \langle j,r\rangle\rangle \in \mathscr{A}$ for $\langle i,l\rangle, \langle j,r\rangle \in \mathscr{V} : i \neq j$ iff customer i can be served at location l before customer j is served at location r by the same vehicle k. Each arc is associated with a cost value $c^{\text{travel}}_{l,r}$ and a time value $t^{\text{travel}}_{l,r}$. Note that for two vertices $v_{i,l}$ and $v_{j,r}$, only locations l and r are relevant to determine the travel cost and travel time between the vertices.

Let $\mathscr{S} \subseteq \mathscr{V}$ be a subset of the vertex set. The *in-arcs*, having their head node in $\mathscr{S}$, are defined as $\delta^-(\mathscr{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathscr{A} : v_{i,l} \notin \mathscr{S}, v_{j,r} \in \mathscr{S}\}$ and the *out-arcs*, having their tail node in $\mathscr{S}$, as $\delta^+(\mathscr{S}) = \{\langle v_{i,l}, v_{j,r}\rangle \in \mathscr{A} : v_{i,l} \in \mathscr{S}, v_{j,r} \notin \mathscr{S}\}$. Singleton sets $\mathscr{S} = \{v_{i,l}\}$ are defined as $\delta^{+|-}(v_{i,l}) := \delta^{+|-}(\{v_{i,l}\})$. If $\langle i,l\rangle \in \delta^-(j,r)$, then $\langle j,r\rangle \in \delta^+(i,l)$, meaning if $\langle i,l\rangle$ is a predecessor of $\langle j,r\rangle$, then $\langle j,r\rangle$ is a successor of $\langle i,l\rangle$.

If each customer can only be served at one location, then the graph of the VPRTW-FL is equal to the routing network of the VRPTW. To show the benefit of the VRPTW-FL over the VRPTW, let us consider the graph in Figure 6.2, which shows a routing network for three customers, three service locations, and the depot. The first customer has two possible locations $\mathscr{L}_1 = \{1,2\}$, the second customer has three possible locations $\mathscr{L}_2 = \{1,2,3\}$ and the third customer has one possible location $\mathscr{L}_3 = \{3\}$.[24] The time windows $[a_i, b_i]$ are given below the customer-location tuples. Let the travel times equal the travel costs, the service time s_i of each customer is equal to 1, and each customer has a preferred location (marked by bold boxes). Furthermore, if the customer is served in the preferred location, location costs of 0 occur and if the customer is served in another location, location costs of 1 occur. Arcs within the same location have travel costs of 0.

If, as in the VRPTW, only one location per customer exists, i.e., for the VPRTW-FL the customers must be served in their preferred location, a vehicle can either serve customers i_1 and i_3, or i_2 and i_3 but never customers i_1 and i_2. Thus, two vehicles are required leading to a total travel time of 14. In the VRPTW-FL, however, serving customer i_2 at his/her alternative location 1 guarantees that one vehicle can serve all customers within a travel time of 10 and additional location swapping cost of 1.

In general, the infeasibility of a VRPTW-FL implies the infeasibility of the corresponding VRPTW, but not vice versa. Moreover, the objective function of an optimal solution of the

[24] Note that edges between the customer-location tuples are needed to track the sequence in which customers are served. Tracking would not be possible in a network only consisting of the locations.

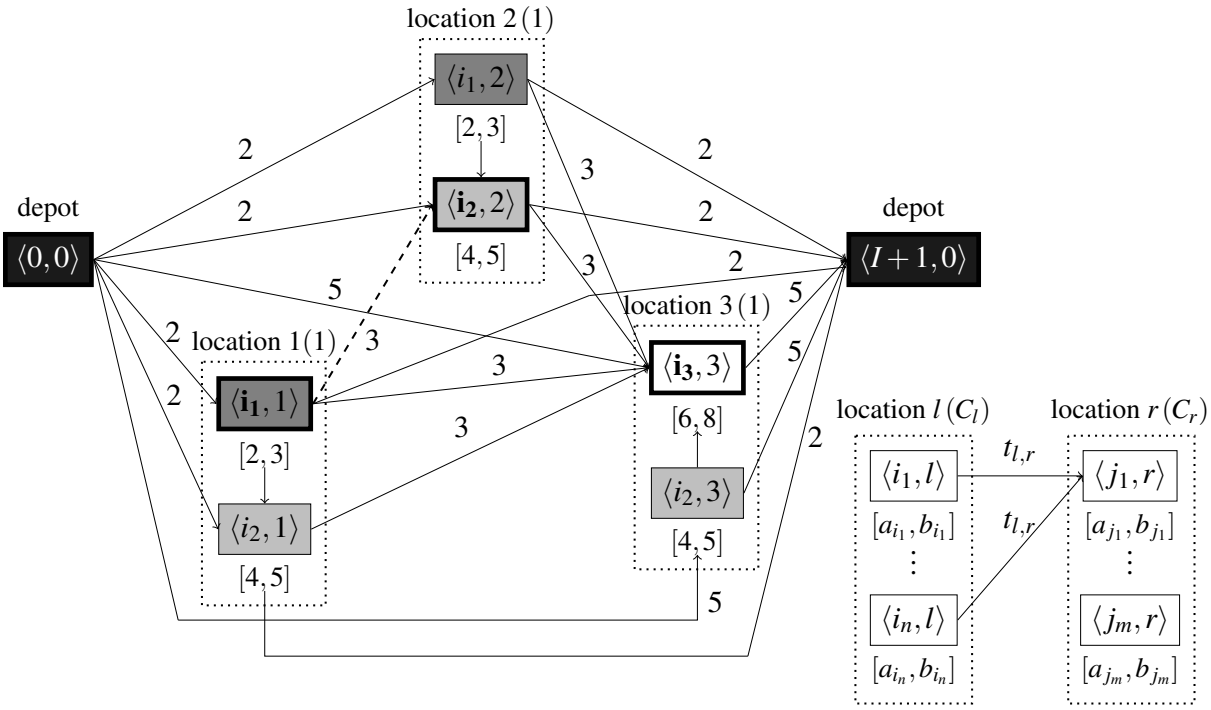


Figure 6.2: *Routing network example: VRPTW vs. VRPTW-FL. Dotted boxes denote locations and include all customers that can be served at this location. Nodes belonging to the same customer are printed in the same color. Bold boxes denote that this location is the customer's preferred location. The dashed arrow displays the connection that is impossible due to time window restrictions. Service times are not displayed as $s_i = 1$ for all $i \in \mathscr{I}$.*

VRPTW yields an upper bound for the VRPTW-FL. To formalize this, function $\eta : \mathscr{I} \times \mathscr{L} \mapsto \mathscr{V}$ is introduced for mapping the swap of customer i from the preferred location to another location at cost $c_{i,l}^{\text{location}}$. Then, a VRPTW-FL instance is uniquely given by tuple $\langle \mathscr{K}, \mathscr{I}, \mathscr{V}, \eta \rangle$ and Theorem 1 holds.

Theorem 1. *Having instances $\tau_1 = \langle \mathscr{K}, \mathscr{I}, \mathscr{V}, \eta \rangle$ and $\tau_2 = \langle \mathscr{K}, \mathscr{I}, \mathscr{V}', \eta \rangle$, which only differ in the vertex sets $\mathscr{V}$ and $\mathscr{V}'$ defined by the customer-location combinations, let z_1^* and z_2^* be the optimal solution for instance τ_1 and τ_2, respectively. If $\mathscr{V}_i' \subseteq \mathscr{V}_i$ holds for all $\mathscr{V}_i' \in \mathscr{V}'$ and $\mathscr{V}_i \in \mathscr{V}$ with $i \in \mathscr{I}$, then $z_2^* \geq z_1^*$.*

If each customer can be assigned to any location, i.e., all location costs $c_{i,l}^{\text{location}}$ are 0 and each location is uncapacitated, then the VRPTW-FL becomes an easy problem because all customers can be served at the location closest to the depot. However, if the location costs are greater than 0 or the locations' capacities are bounded by at least $I - 1$, the VRPTW-FL is *NP*-hard.

6.2.3.3 Mathematical Model

For the VRPTW-FL, we have two main decision variables: $x_{i,l,j,r,k} = 1$, if vehicle $k \in \mathscr{K}$ serves customer $i \in \mathscr{I}$ in location $l \in \mathscr{L}_i$ immediately before serving customer $j \in \mathscr{I}$ in location $r \in \mathscr{L}_j$, and 0 otherwise, and $T_{i,l,k}$ being the start time of serving customer $i \in \mathscr{I}$ in location $l \in \mathscr{L}_i$ by vehicle $k \in \mathscr{K}$. Binary variables $x_{i,l,j,r,k}$ constitute the tours, while continuous variables $T_{i,l,k}$ yield the scheduling decisions.

To account for heterogeneous vehicles, sets and parameters corresponding to a certain vehicle are indexed by k, e.g., $\mathscr{V}_k$ are all vertices which can be reached by vehicle k. The subset of vehicles which can serve a customer i are denoted as $\mathscr{K}_i$. Let $\mathscr{P}$ define the precedence relations between two customers $\langle i, j \rangle$. Customer j can only be served if customer i has already been served.

The location capacity and their utilization is modeled based on the continuous time formulation of the resource-constrained project scheduling problem (cf. [7, 115]). Binary variable $y_{i,l,j,r} = 1$, if service of customer i in location l precedes service of customer j in location r. Flow variable $f_{i,j,l}$ states the number of resource units of location $l \in \mathscr{L}^{\text{bounded}}$ that flow from customer i to customer j, i.e., if i is served right before j in location l. Q_l defines the available number of resource units in location l. For modeling purposes, it is distinguished between two sets of pairs of customer-location-tuples $\langle\langle i,l\rangle,\langle j,r\rangle\rangle$: (1) set $\mathscr{W}$ containing all possible pairs, and (2) set $\mathscr{U}$ for which we know that $\langle i,l\rangle$ must start before $\langle j,r\rangle$.

$$\mathscr{W} = \{\langle\langle i,l\rangle,\langle j,r\rangle\rangle \in \{\langle i,l\rangle \in \mathscr{V} \setminus \langle I+1,0\rangle\} \times \langle j,r\rangle \in \delta^+(i,l)\} \tag{6.1}$$

$$\mathscr{U} = \mathscr{P} \cup \{\langle\langle i,l\rangle,\langle j,r\rangle\rangle \in \mathscr{W} \,|\, b_{i,l} \leq a_{j,r}\} \tag{6.2}$$

Further, let $\mathscr{W}_l \subseteq \mathscr{W}$ be the subset for which the location component of both tuples in the pairs corresponds to l. The VRPTW-FL can now be stated as model (6.3)-(6.24):

$$\min \sum_{k\in\mathscr{K}} \sum_{\langle i,l\rangle\in\mathscr{V}_k\setminus\langle I+1,0\rangle} \sum_{\langle j,r\rangle\in\delta_k^+(i,l)} \left(c^{\text{travel}}_{l,r} + c^{\text{location}}_{j,r}\right) \cdot x_{i,l,j,r,k} \tag{6.3}$$

subject to

$$\sum_{k\in\mathscr{K}_i} \sum_{\langle i,l\rangle\in\mathscr{V}_k} \sum_{\langle j,r\rangle\in\delta_k^+(i,l)} x_{i,l,j,r,k} = 1 \quad \forall i \in \mathscr{I} \tag{6.4}$$

$$\sum_{\langle j,r\rangle\in\delta_k^+(0,0)} x_{0,0,j,r,k} = 1 \quad \forall k \in \mathscr{K} \tag{6.5}$$

$$\sum_{\langle i,l\rangle\in\delta_k^-(j,r)} x_{i,l,j,r,k} - \sum_{\langle i,l\rangle\in\delta_k^+(j,r)} x_{j,r,i,l,k} = 0 \quad \forall k \in \mathscr{K}, \langle j,r\rangle \in \mathscr{V}_k \tag{6.6}$$

$$\sum_{\langle i,l\rangle \in \delta_k^-(I+1,0)} x_{i,l,I+1,0,k} = 1 \quad \forall k \in \mathscr{K} \tag{6.7}$$

$$a_{i,l} \leq T_{i,l,k} \leq b_{i,l} \quad \forall k \in \mathscr{K}, \langle i,l\rangle \in \mathscr{V}_k \tag{6.8}$$

$$T_{i,l,k} + s_i + t_{r,l}^{\text{travel}} - T_{j,r,k} \leq (1 - x_{i,l,j,r,k}) \cdot M_{i,l,j,r} \quad \forall k \in \mathscr{K}, \langle i,l\rangle \in \mathscr{V}_k \backslash \langle I+1,0\rangle, \langle j,r\rangle \in \delta_k^+(i,l) \tag{6.9}$$

$$\sum_{(i,l)\in\mathscr{V}_k} q_i \sum_{(j,r)\in\delta_k^+(i,l)} x_{i,l,j,r,k} \leq Q_k \quad \forall k \in \mathscr{K} \tag{6.10}$$

$$T_{i,l,k_1} + s_i + t_{r,l}^{\text{travel}} \leq T_{j,r,k_2} \quad \forall \langle i,j\rangle \in \mathscr{P}, l \in \mathscr{L}_i, r \in \mathscr{L}_j, k_1 \in \mathscr{K}_i, k_2 \in \mathscr{K}_j \tag{6.11}$$

$$y_{i,l,j,r} = 1 \quad \forall \langle \langle i,l\rangle, \langle j,r\rangle \rangle \in \mathscr{U} \tag{6.12}$$

$$T_{i,l,k_1} - T_{j,r,k_2} \leq (1 - y_{i,l,j,r}) \cdot M_{i,l,j,r} \quad \forall \langle \langle i,l\rangle, \langle j,r\rangle \rangle \in \mathscr{W} \backslash \mathscr{U}, k_1 \in \mathscr{K}_i, k_2 \in \mathscr{K}_j \tag{6.13}$$

$$T_{i,l,k_1} - T_{j,r,k_2} \leq 0 \quad \forall \langle \langle i,l\rangle, \langle j,r\rangle \rangle \in \mathscr{U}, k_1 \in \mathscr{K}_i, k_2 \in \mathscr{K}_j \tag{6.14}$$

$$f_{i,j,l} - y_{i,l,j,l} \leq 0 \quad \forall l \in \mathscr{L}^{\text{bounded}}, \langle i,j\rangle \in \mathscr{W}_l \backslash \langle \langle 0,0\rangle, \langle I+1,0\rangle \rangle \tag{6.15}$$

$$\sum_{\substack{j\in\mathscr{I}_l\cup\langle I+1,0\rangle: \\ i\neq j}} f_{i,j,l} = \sum_{k\in\mathscr{K}} \sum_{\langle j,r\rangle\in\delta_k^+(i,l)} x_{i,l,j,r,k} \quad \forall l \in \mathscr{L}^{\text{bounded}}, i \in \mathscr{I}_l \tag{6.16}$$

$$\sum_{\substack{i\in\mathscr{I}_l\cup\langle 0,0\rangle: \\ i\neq j}} f_{i,j,r} = \sum_{k\in\mathscr{K}} \sum_{\langle i,l\rangle\in\delta_k^-(j,r)} x_{i,l,j,r,k} \quad \forall r \in \mathscr{L}^{\text{bounded}}, j \in \mathscr{I}_r \tag{6.17}$$

$$\sum_{j\in\mathscr{I}_l\cup\langle I+1,0\rangle} f_{0,j,l} = Q_l \quad \forall l \in \mathscr{L}^{\text{bounded}} \tag{6.18}$$

$$\sum_{i\in\mathscr{I}_l\cup\langle 0,0\rangle} f_{i,I+1,l} = Q_l \quad \forall l \in \mathscr{L}^{\text{bounded}} \tag{6.19}$$

$$f_{I+1,0,l} = Q_l \quad \forall l \in \mathscr{L}^{\text{bounded}} \tag{6.20}$$

$$x_{i,l,j,r,k} \in \{0,1\} \quad \forall k \in \mathscr{K}, \langle i,l\rangle \in \mathscr{V}_k, \langle j,r\rangle \in \delta_k^+(i,l) \tag{6.21}$$

$$T_{i,l,k} \geq 0 \quad \forall k \in \mathscr{K}, \langle i,l\rangle \in \mathscr{V}_k \tag{6.22}$$

$$y_{i,l,j,r} \in \{0,1\} \quad \forall \langle \langle i,l\rangle, \langle j,r\rangle \rangle \in \mathscr{W} \tag{6.23}$$

$$0 \leq f_{i,j,l} \leq Q_l \quad \forall l \in \mathscr{L}^{\text{bounded}}, \langle i,j\rangle \in \mathscr{W}_l \tag{6.24}$$

Objective function (6.3) minimizes the sum of travel and location costs. The VRPTW-FL's constraints can be divided into three parts:

Tour constraints (6.4)-(6.7): Constraints (6.4) ensure that every customer is served exactly once. Constraints (6.5)-(6.7) define the tour of each vehicle: constraints (6.5) and (6.7) impose a tour start and end at the depot, while constraints (6.6) are the flow conservation constraints.
Scheduling constraints (6.8)-(6.11): Constraints (6.8) set the start times for serving customer i, while constraints (6.9) set the time difference between two customers served consecutively by the same vehicle by linking variables $x_{i,l,j,r,k}$ and $T_{i,l,k}$. Constraints (6.10) ensure that a vehicle k cannot satisfy more customer demand than its capacity limit Q_k. Constraints (6.11) ensure the precedence relations.
Location capacity constraints (6.12)-(6.20): Constraints (6.12) fix the sequences that are already implied by the possible start times of the customers or the precedence relations. Correct separation times between preceding customer-location tuple $\langle i,l \rangle$ and succeeding tuple $\langle j,r \rangle$ are guaranteed by constraints (6.13) and (6.14). Parameter $M_{i,l,j,r}$ is a sufficiently large constant that can be set to $M_{i,l,j,r} = \max\{b_{i,l} + s_i + t^{\text{travel}}_{l,r} - a_{j,r}, 0\}$. Constraints (6.15) connect the variables $y_{i,l,j,r}$ and $f_{i,j,l}$, and ensure that a resource flow only exists between preceding customer-location tuples. Both the outflow (6.16) and the inflow (6.17) of a resource flow can only exist if the involved customers are served in the respective locations. Constraints (6.18) to (6.19) ensure that the total flow of one resource is equal to the corresponding location capacity.

The variable domains are given in constraints (6.21) to (6.24).

6.2.4 Solution Methodology

When using a standard MIP solver and the compact formulation presented in § 6.2.3.3, only very small problem instances can be solved. To solver larger problems, exact decomposition or heuristic solution approaches could be used. Jungwirth et al. [103] present a branch-price-and-cut algorithm for the VRPTW-FL. However, the algorithm is computational demanding and requires a state-of-the-art commercial MIP solver, which might not be pertinent for practitioners. Therefore, the proposed solution approach is based on an *Adaptive Large Neighborhood Search* (ALNS) framework where the general idea behind the algorithm is easy to grasp and close to optimal solution can be obtained relatively quickly without the need for an additional solver.

The ALNS is a well-established framework for solving routing problems. It was originally developed by [173] and is still used in publications addressing new variants of VRPs (see, e.g., [145, 116, 9, 137, 142, 157, 182]). The ALNS works well for the VRPTW-FL not only due to its good performance for the VRP, but also due to the incorporation of other desirable features such as the simplicity of the underlying concept and its flexibility with respect to VRP variants [125].

The ALNS is an extension of the *Large Neighborhood Search* (LNS) introduced by Shaw [189] and relies on the ruin-and-recreate principle applied in Schrimpf et al. [184], which is similar to the rip-up principle of Dees and Karger [37]. In a first phase, an initial solution is constructed, and then in an improvement phase, the ALNS iteratively destroys parts of this solution using randomly selected destroy operators and reconstructs the destroyed solution with randomly selected repair operators. The combination of destroy and repair operators defines the neighborhood in which the new solution will be sought. If the solution is accepted according to an acceptance criterion, the current solution is replaced by the new solution and the procedure starts again. The probability of selecting a particular destroy and a repair operator is adjusted based on the success (or lack thereof) of improving a temporary solution in the past.

The proposed ALNS incorporates innovative features in both, the construction phase and the improvement phase. In the former, the heuristic follows the k-regret insertion approach of Potvin and Rousseau [163], which is extended with a backtracking mechanism to alter unsatisfactory decisions at an early stage. To counteract infeasible sequences of subsequently planned customers due to the simple nature of the k-regret procedure, the procedure returns to an earlier stage of the insertion with a given probability, and restarts from this stage by inserting another customer.

In the improvement phase, the proposed framework deviates from the standard ALNS presented by [173] in three ways: (1) it allows temporarily infeasible solutions, however, sanction the infeasibilities in the objective function with penalties; (2) it dynamically adjusts these penalties depending on how often certain features have been violated in past iterations; and (3) new operators were developed, which exploit the underlying problem structure of potentially having more than one location per customer.

The generation of temporarily infeasible solutions enables a better traversing of the search space because it reduces the chance of getting stuck in a local optimum [34], and by oscillating between feasible and infeasible regions with the appropriate penalty parameters the border of feasibility is sought, a region which is very promising for finding high quality solutions [75, 209]. For the VRPTW-FL, the framework allows four infeasibilities: (1) unscheduled customers, (2) violations of time windows, (3) violations of precedence relations, and (4) skill violations. In therapist scheduling, these are precisely the four aspects that a human planner would relax when faced with a hard scheduling task, where no feasible solution can be obtained manually.

Updating penalties for the violation terms dynamically extends the self-adaptiveness from operator probability updates to objective function weights and, therefore, makes the approach more flexible and robust for dealing with the problem at hand. From a formal point of view, the proposed approach combines the ALNS with a *Guided Local Search* (GLS) as employed in Voudouris and Tsang [213] and, thus, leads to a hybrid version of the ALNS. A simpler version of such an adaptive mechanism was originally formulated by Cordeau et al. [33] and is, e.g., used in Schiffer and Walther [182].

In what follows, the hybrid ALNS framework is first formalized in § 6.2.4.1. In § 6.2.4.2, the construction heuristic is detailed, including the backtracking mechanism. In § 6.2.4.3, the reader is provided with information about the destroy and repair operators used, and the update procedures for the operators and objective function weights. Finally in § 6.2.4.4, implementation details are provided focusing on preprocessing and parameter optimization. A concise overview of the parameters that are introduced in the following subsections and their corresponding values used in the computational study are given in the addendum § 6.4.2 of this chapter.

6.2.4.1 Formal Hybrid ALNS Framework

Let s be a vector representing any (partial) solution for the VRPTW-FL and let $f(s)$ be the function that returns the objective function value for s as stated in (6.3), then the proposed heuristic works on the modified objective function:

$$\min f^{\text{mod}}(s) = f(s) + \sum_{i \in \mathscr{I}} \sum_{\upsilon \in \{\text{na,tw,pred,skill}\}} (\lambda^{\upsilon} \cdot p_i^{\upsilon} \cdot I_i^{\upsilon}(s)) \,, \tag{6.25}$$

where $I_i^{\text{na}}(s)$, $I_i^{\text{tw}}(s)$, $I_i^{\text{pred}}(s)$, and $I_i^{\text{skill}}(s)$ are indicator functions equal to 1, if in a solution s customer i is not assigned to any vehicle, if the time window of customer i is violated, if the precedence relation of customer i is violated, and if the skill level is violated, respectively.[25] The penalty terms are denoted by p_i^{na}, p_i^{tw}, p_i^{pred}, and p_i^{skill}; see § 6.2.4.3.2 for how penalties are set and updated. The weights of penalties compared to routing and location costs is controlled by λ^{υ}, where $\upsilon \in \{\text{na}, \text{tw}, \text{pred}, \text{skill}\}$.

Algorithm 2 provides the pseudo code for the hybrid ALNS framework. A solution is represented by s, and an initial solution s^{init} from the construction phase serves as input. The initial solution is set equal to the best global optimum found thus far (see Algorithm 2 Line 1).

In the main loop 2 – 19, destroy operator $d \in \Omega^-$ and repair operators $r \in \Omega^+$ modify the current solution s^{current}. In each iteration h of the main loop, $q \geq 1$ pairs of destroy and repair operators are randomly selected to destroy and repair $n \in [\underline{n}_h, \bar{n}_h]$ elements in solution s^{current}. Parameters $\underline{n}_h$ and $\bar{n}_h$ define the lower and upper bounds of affected elements in each iteration h.

The framework randomly selects a pair of destroy and repair operators in Step 3. The destroy and repair operators are described in detail in § 6.2.4.3.3 and § 6.2.4.3.4, respectively. When deriving these operators, a destroyed solution should be repairable by any repair operator. The probability of selecting a destroy operator ρ^- and a repair operator ρ^+ depends on their past success. The update procedure for the probabilities ρ^+ and ρ^- is described in § 6.2.4.3.1.

If the new solution s^{temp} improves the best global solution s^{best}, the framework updates s^{best} and the current solution s^{current} (see Lines 5 - 6). Otherwise, the algorithm checks whether the temporary solution s^{temp} is accepted as a new searching point using some criteria defined by the

[25] Precedence violation of customer i is defined as service of i has started although service of preceding customer j has not been finished yet.

local search framework (see Lines 8 - 16). In the proposed method, a *Simulated Annealing* (SA) framework (see Kirkpatrick et al. [113]) is used, which defines the acceptance of the solution and the direction of the destroy and repair operators.

The structure of Algorithm 2 is based on Pisinger and Ropke [161]. However, the main difference is Line 18, where the objective penalty terms p^{na}, p^{tw}, p^{pred}, and p^{skill} are updated (see § 6.2.4.3.2 for a detailed description). The algorithm terminates after a stopping criteria has been met, e.g., a total number of iterations or iterations without improvement, then the best global solution s^{best} is returned.

Algorithm 2: HYBRID ADAPTIVE LARGE NEIGHBORHOOD SEARCH

input : initial solution s^{init} with objective $f^{\text{mod}}(s^{\text{init}})$; `// (see § 6.2.4.2)`

1 $s^{\text{best}} = s^{\text{current}} = s^{\text{init}}$, $\rho^-(1,\ldots,1)$, $\rho^+(1,\ldots,1)$;
2 **while** *stopping criteria is not met* **do**
3 select a pair of destroy and repair operators $d \in \Omega^-$ and $r \in \Omega^+$ based on ρ^- and ρ^+ ; `// (see § 6.2.4.3.3 and § 6.2.4.3.4)`
4 $s^{\text{temp}} = r(d(s^{\text{current}}))$;
5 **if** $f^{simple}(s^{best}) < f^{simple}(s^{temp})$ **then** ▷ cf. Eq. 6.27
6 $s^{\text{best}} = s^{\text{current}} = s^{\text{temp}}$;
7 updatePi (d, r, σ_1);
8 **else if** s^{temp} *not visited* $\wedge$ $f^{mod}(s^{temp}) < f^{mod}(s^{current})$ **then** ▷ cf. Eq. 6.25
9 $s^{\text{current}} = s^{\text{temp}}$;
10 updatePi (d, r, σ_2);
11 **else if** s^{temp} *not visited* $\wedge$ (s^{temp} *is accepted)* **then**
12 $s^{\text{current}} = s^{\text{temp}}$;
13 updatePi (d, r, σ_3);
14 **else**
15 // No new solution found ;
16 **end if**
17 update ρ^- and ρ^+ according to Eq. 6.29 ; `// (see § 6.2.4.3.1)`
18 update objective penalty terms p^{na}, p^{tw}, p^{pred}, and p^{skill} ; `// (see § 6.2.4.3.2)`
19 **end while**
20 **return** s^{best}

Figure 6.3 shows exemplary the relationship between the costs for $s^{\text{best}}, s^{\text{current}}$, and s^{temp} for a hospital instance with $n = 40$ customers and $25'000$ iterations. The green skyline denotes the costs for s^{best}, the blue line shows the costs for s^{current} which can still be accepted due to simulated annealing, and gray dots represents the costs for s^{temp}, i.e., the costs after each iteration.

In Figure 6.4, the behavior of the simulated annealing w.r.t. the temperature value is shown. The chance by accepting a solution – even though it shows worse cost values – decreases over time. The parameter settings for the simulated annealing are reported in the addendum § 6.4.2.

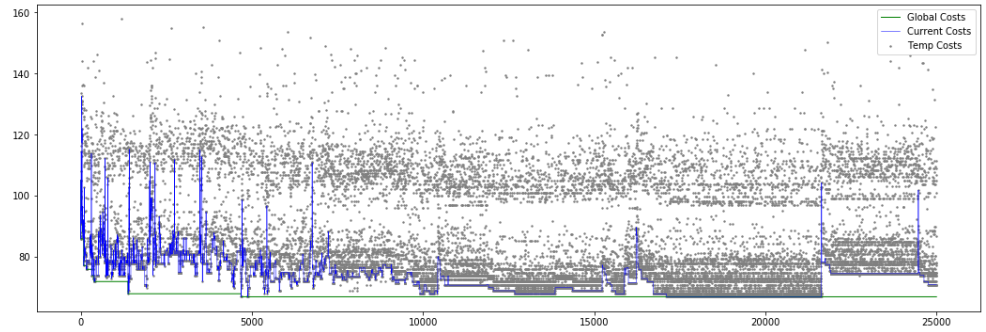


Figure 6.3: *Example of ALNS on hospital instance with n=40 customers for 25k iterations showing the relationship between s^{best}, $s^{current}$, and s^{temp} .*

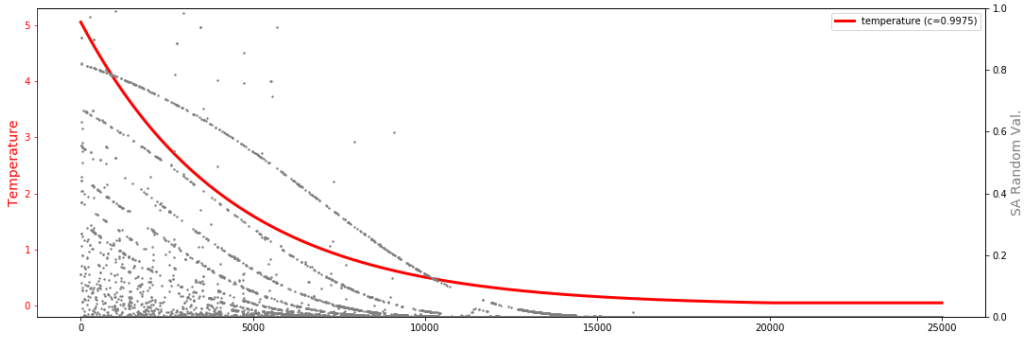


Figure 6.4: *Example of the simulated annealing used for the ALNS run shown in Figure 6.3*

6.2.4.2 Construction Phase

The construction heuristic is similar to the κ-regret[26] approach of Potvin and Rousseau [163], which can be seen as a greedy based insertion heuristic with a look ahead perspective (see Algorithm 3). In the VRPTW-FL, the look ahead perspective becomes even more crucial than in the VRPTW as a good assignment of customers to vehicle routes may still be infeasible due to a poor assignment of customers to locations.

A route r_k for each vehicle $k \in \mathcal{K}$ is represented by an ordered sequence

$$r_k = \big[\langle i_0, l_0, T_0\rangle, \ldots, \langle i_{m_k-1}, l_{m_k-1}, T_{m_k-1}\rangle, \langle i_{m_k}, l_{m_k}, T_{m_k}\rangle, \\ \langle i_{m_k+1}, l_{m_k+1}, T_{m_k+1}\rangle, \ldots \langle i_{n_k}, l_{n_k}, T_{n_k}\rangle\big]$$

of customer-location-start time tuples with $\langle i_{m_k}, l_{m_k}\rangle \in \mathcal{V}$ and $T_{m_k} \in [a_{i_m}, b_{i_m}]$. The customers in each route r_k are served according to the order given in the route sequence, i.e., for each position

[26] To avoid confusing indices k for vehicles and the k-regret approach, index κ is used for the k-regret approach.

$0 \leq m_k \leq n_k$ in route r_k we have:

$$T_{m_k-1} + s_{i_{m_k-1}} + t^{\text{travel}}_{l_{m_k-1},l_{m_k}} \leq T_{m_k}. \tag{6.26}$$

At the beginning of the construction heuristic, each vehicle route r_k contains only tuples for starting and ending the tour in the depot, i.e., $\langle i_0, l_0, T_0 \rangle = \langle 0,0,0 \rangle$ and $\langle i_{n_k}, l_{n_k}, T_{n_k} \rangle = \langle n+1,0,T \rangle$, respectively.

The goal of the heuristic is to sequentially insert one customer-location-start time tuple in one position of one of the $|\mathscr{K}|$ routes (one route for each vehicle) such that the capacities of the locations are satisfied and Inequality (6.26) holds. However, instead of selecting the best greedy-based position within the routes, the next route position yielding the highest regret between the 1-st and the κ-th best insertion position between all routes is selected. A large gap between the best and the κ-st position indicates that a later assignment might be difficult or infeasible. The difference between the proposed regret approach and Potvin and Rousseau [163] is that the κ best insertion positions over all routes is considered while Potvin and Rousseau [163] consider the κ best routes to insert a customer. The proposed construction heuristic works on a simplification of objective function (6.25) where the penalty values (costs) for each type of violation are fixed and equal for each customer:

$$\min f^{\text{simple}}(s) = f(s) + \lambda \cdot \sum_{i \in \mathscr{I}} \sum_{\upsilon \in \{na,tw,pred,skill\}} (c^{\upsilon} \cdot I_i^{\upsilon}(s)) \tag{6.27}$$

Let $\mathscr{R}$ be the set of all routes over all vehicles $k \in \mathscr{K}$ and let $g_\kappa(i,\mathscr{R})$ denote the objective function value, if customer i is inserted in the κ-th best position of all routes $r_k \in \mathscr{R}$, i.e., we have $g_\kappa(i,\mathscr{R}) \leq g_{\kappa+1}(i,\mathscr{R})$ for all $r_k \in \mathscr{R}$. For objective function value $g_\kappa(i,\mathscr{R})$, let us denote by $l(g_\kappa(i,\mathscr{R}))$, $T(g_\kappa(i,\mathscr{R}))$ and $m(g_\kappa(i,\mathscr{R}))$ the corresponding location, start time, and insertion position of customer i in routes $\mathscr{R}$. Let $g_\kappa(i,\mathscr{R}) = \infty$ for all $\kappa \geq 2$ if only one possible insertion position is left for customer i, i.e., customer i can only be assigned and scheduled in one location and in one route at one insertion position. Let $\mathscr{I}^{\text{na}}$ be the subset of customers, which have not yet been assigned to one route. For all routes $r_k \in \mathscr{R}$ and customer $i \in \mathscr{I}^{\text{na}}$ the following regret measure is computed:

$$\Delta g_\kappa(i,\mathscr{R}) = g_\kappa(i,\mathscr{R}) - g_1(i,\mathscr{R}). \tag{6.28}$$

Measure $\Delta g_\kappa(i,\mathscr{R})$ yields the difference between the best insertion position for a customer i and its κ-th best insertion position with respect to all routes. The regret for a customer indicates what can be lost in later insertions, if the customer is not immediately inserted in the best insertion position. A large regret measure indicates that the number of interesting alternative positions for inserting the customer is small and, thus, this customer should be considered first. On the other hand, a small regret measure indicates that the customer can easily be inserted into alternative positions in later iterations without losing much. The customer and vehicle with the greatest

regret measure is given by customer-route combination $\langle i^*, r_{k^*} \rangle = \arg\max_{i \in \mathscr{I}^{\text{na}}} \{\Delta g_\kappa(i, \mathscr{R})\}$. Thus, customer i^* is inserted in route $r_{k^*} \in \mathscr{R}$ at position $m(g_1(i^*, r_{k^*}))$; the service of customer i^* starts at time $T(g_1(i^*, r_{k^*}))$ at location $l(g_1(i^* r_{k^*}))$. If customer i cannot be inserted into any route, the regret measure $\Delta g_1(i, \mathscr{R})$ is 0 as we define that $\infty - \infty = 0$. This is either the result of a bad insertion of one or several customers in previous iterations or the instance is generally infeasible.

Let us assume that a feasible solution exists. Then, current infeasibility originates either because no insertion position exists such that Inequality (6.26) holds or because no location is available for customer i. To provide a repair mechanism, a backtracking-branching procedure is implemented.

Algorithm 3 illustrates the different steps of the constructive heuristic with backtracking. The initial solution s_0, only containing the depot nodes, is added to the solutions set $\mathscr{S}$, which contains all partial solutions that could not be pruned due to infeasibility (see Algorithm 3 Lines 1-2). The algorithm randomly removes a customer i_1 from the set of not yet assigned customers $\mathscr{I}^{\text{na}}$, and adds customer i_1 who will be served in the preferred location l_1 to route r_{k_0} (see Lines 3-7). To increase diversity and, thus, to find potentially better solutions, the entire heuristic is started multiple times (Line 3) and the subsequent steps are performed for several κ values (Line 8).

While not all customers have been assigned, the regret measure for inserting every remaining customer $i \in \mathscr{I}^{\text{na}}$ in partial solution s_h is calculated (Lines 9-11). If for all remaining customers a positive regret measure exists, i.e., every customer can be inserted in the partial solution s_h, the best insertion position is determined. The set of not yet assigned customers $\mathscr{I}^{\text{na}}$ and the solution set $\mathscr{S}$ are then updated (Lines 12-16).

However, if for at least one customer no positive regret measure exists, i.e., this customer cannot be inserted in the partial solution, this solution becomes infeasible. The procedure then removes s_h, the partial solution which was earlier added to the set of partial solutions $\mathscr{S}$, and returns to $\text{pred}(s_h)$, the predecessor of s_h. To proceed to another solution from $\text{pred}(s_h)$, the deleted customer-location-start time tuple and the corresponding route r_k is stored as tuple $\langle i_h, l_h, T_h, r_{k_h} \rangle$ in a list of forbidden insertions $\mathscr{F}(\text{pred}(s_h))$ of the partial solution $\text{pred}(s_h)$ (Lines 17-20). The next insertion will then be the best customer-route combination with respect to regret measure (6.28) such that the corresponding customer-location start time route tuple is not contained on forbidden list $\mathscr{F}(\text{pred}(s_h))$, i.e., $(i^*, r_k^*) = \arg\max_{i \in \mathscr{I}^{\text{na}}} \big\{\Delta g_\kappa(i, \mathscr{R}) \mid (i, l(g_\kappa(i, \mathscr{R})), T(g_\kappa(i, \mathscr{R})), \mathscr{R}) \notin \mathscr{F}(\text{pred}(s_h))\big\}$.

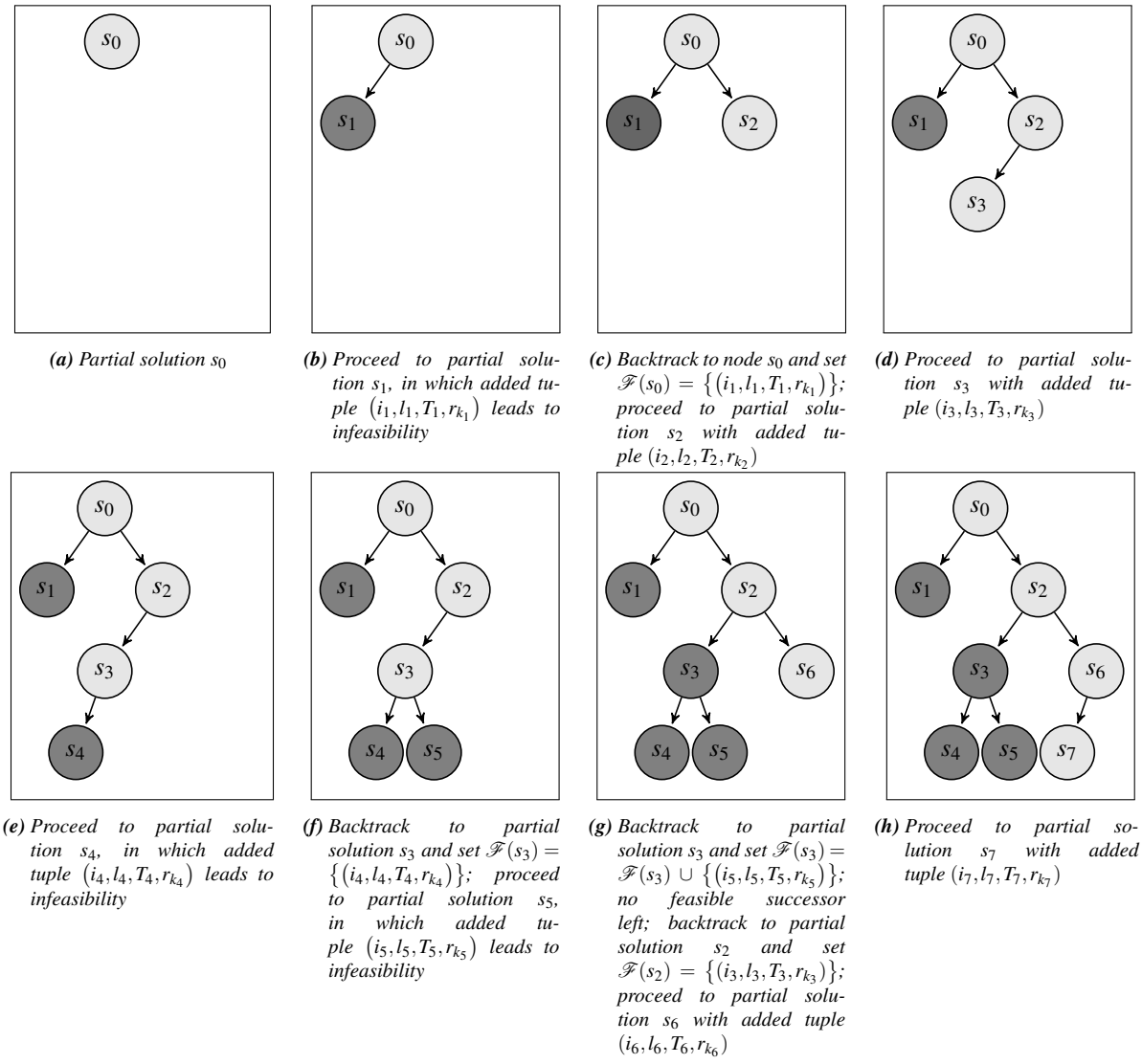
If again no feasible successor exists, i.e., the regret measure is 0 for at least one customer, the procedure returns to $\text{pred}(s_h)$'s predecessor, for which the corresponding customer-location start time route tuple is forbidden which would lead to $\text{pred}(s_h)$ again. Therefore, the procedure generates a search tree in a depth-first search manner.

Finally, solution s^{best} having the minimal objective function value is returned (Line 26). A graphical example of the backtracking mechanism is provided in Figure 6.5.

Algorithm 3: CONSTRUCTION PHASE

Initialize partial solution s_0 with $r_k = [\langle 0,0,0\rangle, \langle n+1,0,T\rangle]\ \forall k \in \mathscr{K}$; $\mathscr{I}^{\text{na}} = \mathscr{I}$;
set $\mathscr{S} = \{s_0\}$;
while *max number of restarts not reached* **do**
  randomly select $i \in \mathscr{I}^{\text{na}}$; set $\mathscr{I}^{\text{na}} = \mathscr{I}^{\text{na}} \setminus \{i\}$;
  if $\Delta g_\kappa(i,\mathscr{R}) > 0$ **then**
    $s_1 \leftarrow$ add tuple $\langle i_1, l_1(g(i,\mathscr{R}), t_1(g(i,\mathscr{R}))\rangle$ to position $m_1(g(i,\mathscr{R}))$ in route $r_{k_0} \in \mathscr{R}$ of s_0;
    set $\mathscr{S} = \mathscr{S} \cup \{s_1\}$;
    foreach *regret* κ **do**
      while $\mathscr{I}^{na} \neq \emptyset$ **do**
        $s_h \leftarrow$ select last inserted partial solution in $\mathscr{S}$;
        compute regret measure for all not inserted customers ;
        if $\Delta g_\kappa(i,\mathscr{R}) > 0\ \forall i \in \mathscr{I}^{na}$ **then**
          $(i^*, r_k^*) \leftarrow \arg\max_{i \in \mathscr{I}^{\text{na}}} \{\Delta g_\kappa(i,\mathscr{R})\}$;
          $s_{h+1} \leftarrow$ add tuple $\langle i_{h+1}^*, l_{h+1}(g(i^*, r_k^*)), t_{h+1}(g(i^*, r_k^*))\rangle$ to position $m_{h+1}(g(i^*, r_k^*))$ in route $r_{k_{h+1}}^*$, i.e. $r_k^* = \left[\langle 0,0,0\rangle, \ldots, \langle i^*, l(g(i^*, r_k^*)), t(g(i^*, r_k^*))\rangle, \ldots, \langle n+1,0,T\rangle\right]$
          $\mathscr{I}^{\text{na}} = \mathscr{I}^{\text{na}} \setminus \{i^*\}$;
          set $\mathscr{S} = \mathscr{S} \cup \{s_{h+1}\}$;
        else
          set $\mathscr{S} = \mathscr{S} \setminus \{s_h\}$;
          return to predecessor of pred(s_h) ;
          set predecessor's forbidden list $\mathscr{F}(\text{pred}(s_h)) = \mathscr{F}(\text{pred}(s_h)) \cup \{\langle i_h, l_h, T_h, r_{k_h}\rangle\}$;
        end if
      end while
    end foreach
  end if
end while
$s^{\text{best}} \leftarrow \text{argmin}_{s \in \mathscr{S}} \{f^{\text{mod}}(s)\}$;
return s^{best}

During the first computational tests, the following observation was made: Returning to the direct predecessor of an infeasible partial solution does generally not correct the solution as desired, especially, if only a few customers are left to insert. Many iterations of backtracking are needed, until a feasible solution is found. The reason is that an insertion influences the insertion position of every subsequently inserted customer. Thus, customers being inserted earlier have greater influence on the structure of the solution than later ones. If poor insertion decisions have



(a) *Partial solution s_0*

(b) *Proceed to partial solution s_1, in which added tuple (i_1, l_1, T_1, r_{k_1}) leads to infeasibility*

(c) *Backtrack to node s_0 and set $\mathscr{F}(s_0) = \{(i_1, l_1, T_1, r_{k_1})\}$; proceed to partial solution s_2 with added tuple (i_2, l_2, T_2, r_{k_2})*

(d) *Proceed to partial solution s_3 with added tuple (i_3, l_3, T_3, r_{k_3})*

(e) *Proceed to partial solution s_4, in which added tuple (i_4, l_4, T_4, r_{k_4}) leads to infeasibility*

(f) *Backtrack to partial solution s_3 and set $\mathscr{F}(s_3) = \{(i_4, l_4, T_4, r_{k_4})\}$; proceed to partial solution s_5, in which added tuple (i_5, l_5, T_5, r_{k_5}) leads to infeasibility*

(g) *Backtrack to partial solution s_3 and set $\mathscr{F}(s_3) = \mathscr{F}(s_3) \cup \{(i_5, l_5, T_5, r_{k_5})\}$; no feasible successor left; backtrack to partial solution s_2 and set $\mathscr{F}(s_2) = \{(i_3, l_3, T_3, r_{k_3})\}$; proceed to partial solution s_6 with added tuple (i_6, l_6, T_6, r_{k_6})*

(h) *Proceed to partial solution s_7 with added tuple (i_7, l_7, T_7, r_{k_7})*

Figure 6.5: *Example for backtracking in the constructive phase*

been made early, it is unlikely to correct these tens of iterations later by backtracking. Therefore, once a partial solution becomes infeasible, the procedure does not backtrack to its immediate predecessor but to one of the first n, e.g., $n = 5$, customers inserted. By doing so, high quality solutions are generated while saving much computational time, compared to a full depth-first search.

6.2.4.3 Operators and Update Functions

The algorithmic behavior of an ALNS depends heavily on (a) the destroy operators Ω^- and repair operators Ω^+ employed, i.e., the neighborhoods that can be searched, and (b) the updates of the operator weights, i.e., how fast the ALNS adjusts the probabilities of selecting a certain operator.

In the proposed hybrid ALNS, the updates of the penalty terms in the objective function also play a crucial role. In this section, it is described how the update procedures work and what operators are used. The focus lies on newly developed operators employing specific properties of the VRPTW-FL, such as multiple locations.

6.2.4.3.1 Update Operator Weights

To adjust the likelihood of selecting a specific operator, the framework follows the approach of [173]. Initially all operators $j \in \Omega^{+|-}$ get assigned the same weight w_j, e.g., 1, and the probability ρ_j of selecting an operator j is:

$$\rho_j = \frac{w_j}{\sum_{i=1}^{|\Omega|} w_i} \tag{6.29}$$

For a given number of iterations, the success of the operators is measured by a score π_j with $j \in \Omega^{+|-}$. Four cases are distinguished:

1. if a new global best solution is found, the score is raised by σ_1;
2. if a new and not yet visited solution is found with a better objective function value than the current solution, the score is raised by $\sigma_2 < \sigma_1$;
3. if a new and unvisited solution did not improve the current solution but is still accepted, the score is raised by $\sigma_3 < \sigma_2$;
4. if a solution is found, but this solution has already been visited in prior iterations, the score remains unchanged.

Once a certain number of iterations has been reached, the operator weights are updated according to the recorded scores π_j and the counter θ_j (cf. Equation (6.30)). The counter θ_j measures how often the operator has been applied:

$$w_j^{\text{updated}} = w_j \cdot (1-r) + r \cdot \frac{\pi_j}{\theta_j} \tag{6.30}$$

Reaction factor r controls how fast the weights adapt to the success in the last iterations.

6.2.4.3.2 Update Objective Penalty Terms

In the augmented cost function (6.25), penalty terms are used to penalize feasibility violations. These penalties are dynamically adjusted depending on the frequency of the violation in the past and the severity of the violations. This approach of dynamically adjusting objective function weights follows the GLS employed in Voudouris and Tsang [213] and will be described in the following.

Let f_j be a specific feature, e.g., the non-assignment of customer i_1, and indicator function $I_j(s)$ is 1, if solution s has feature f_j, and 0 otherwise. Each feature f_j, i.e., the violation of a specific constraint, is associated with a constant cost value c_j and a dynamically adjusting penalty value p_j for the objective function. All penalty values are set to 0 initially, i.e., $p_i^{\text{na}} = p_i^{\text{tw}} = p_i^{\text{pred}} = p_i^{\text{skill}} = 0 \ \ \forall i \in \mathscr{I}$. After a certain number of iterations, the penalty values are updated for a predefined number of features yielding the highest utility value as defined in Equation (6.31), where s_h is the current solution at iteration h and $n_i^{(j)}$ denotes the number of occurrences of the j-th penalty for customer i since the last update.

$$u_i(s_h, f^{(j)}) = I_i^{(j)}(s_h) \cdot \frac{n_i^{(j)} c^{(j)}}{1 + p_i^{(j)}} \tag{6.31}$$

The utility function is used because (a) updating all violated features equally would not change the direction of the search and lead to very similar solutions, and (b) updating only the penalties of features with the highest cost would bias the algorithm towards penalizing high cost features. The denominator $1 + p_j$ counteracts the latter since an increasing penalty p_j reduces the utility value. Note that while Voudouris and Tsang [213] update the penalties once the heuristic is stuck in a local minimum, the proposed framework updates the penalties after a certain number of iterations, which is similar to updating the operator weights in an ALNS (cf. §6.2.4.3.1).

6.2.4.3.3 Destroy Operators

A very useful property of the ALNS is that it can incorporate a multitude of neighborhoods to address specific characteristics of the problem at hand and, thus, a multitude of destroy and repair operators have been developed (see [116] for a good overview). In this section, the destroy operators being applied are described, and in the subsequent section the repair operators being applied. As the VRPTW-FL is a generalization of the VRPTW (see §6.2.3.2), all operators are also applicable for the VRPTW. The effectiveness of the procedures will be shown in the computational study in § 6.2.5.2.3. The operators used are largely taken from the literature, and adapted to the problem setting with flexible delivery locations. Furthermore, the thesis presents seven additional operators specifically designed to deal with flexible delivery locations. The operators taken from the literature and the corresponding sources are: random destroy, worst destroy, simplified Shaw (proximity) destroy, cluster destroy, time related destroy, history based destroy (neighbor graph destroy, request graph destroy) [173, 174, 160], related (Shaw) destroy [189], and random route destroy [142].

For the VRPTW-FL, the customer-locations are very important. Therefore, the thesis introduces four operators specifically addressing the spatial arrangement of service locations: location related destroy, cluster k-means destroy, zone destroy, and subroute destroy. In addition, a modified time related destroy is used and the thesis introduces a start time flexibility destroy. For all operators incorporating some kind of relatedness, the algorithm first removes one customer-

location tuple randomly and then determines the relatedness with regards to this tuple to remove further tuples.

Time related destroy In the time-related destroy operator, those customers i and j are selected which have a strong relation to each other with respect to possible service times. The relatedness $D^{\text{time}}(i,j)$ between two customers is measured as follows:

$$D^{\text{time}}(i,j) = \frac{T}{\left(\alpha_1 \cdot \bar{T}_{i,j} + \alpha_2 \cdot |T_i - T_j|\right)}, \tag{6.32}$$

where $\bar{T}_{i,j}$ is the average time difference between all possible start times of i and j:

$$\bar{T}_{i,j} = \left| \frac{a_i + b_i}{2} - \frac{a_j + b_j}{2} \right|, \tag{6.33}$$

and $|T_i - T_j|$ is the time difference between the start times of customers i and j in the current solution. At first, one customer i is removed at random, and then the $n-1$ customers who are most related to i are removed. This logic also applies to the other related destroy operators. Note that in the time related destroy presented in [160] only the term $|T_i - T_j|$ is used to select the removals.

Location related destroy Similar to the time related destroy, this operator removes vertices, which are very similar in terms of their locations (cf. Equation (6.34)). The location relatedness $D^{\text{loc}}(i,j)$ between two customers i and j is the number of common possible service locations divided by the number of locations available for the customer with less location flexibility ($\min\{|\mathscr{L}_i|, |\mathscr{L}_j|\}$).

$$D^{\text{loc}}(i,j) = 1 - \text{sim}^{\text{loc}}(i,j) = 1 - \frac{|\mathscr{L}_i \cap \mathscr{L}_j|}{\min\{|\mathscr{L}_i|, |\mathscr{L}_j|\}} \tag{6.34}$$

Location and time related destroy The weighted combination of the location related destroy and the time related destroy is defined as:

$$D^{\text{loc,time}}(i,j) = \beta_1 \cdot D^{\text{loc}}(i,j) + \beta_2 \cdot D^{\text{time}}(i,j). \tag{6.35}$$

If $\beta_1 = 0$ the operator is equal to the time related destroy, if $\beta_2 = 0$ the operator is equal to the location related destroy.

Cluster destroy k-means While Ropke and Pisinger [174] describe a cluster destroy based on the minimum spanning tree algorithm by Kruskal [118], the thesis introduces a cluster destroy based on the very popular k-means clustering. The goal of k-means clustering is to partition a set into k disjoint subsets, such that the sum of the squared deviations (distances) from the positions x_j of all elements j in the clusters $\mathscr{S}_i$ to the clusters' centers μ_i is minimal. Mathematically,

this is:

$$\min \sum_{i=1}^{k} \sum_{j \in \mathscr{S}_i} \left(x_j - \mu_i\right)^2. \tag{6.36}$$

For a recent overview of clustering algorithms and a more detailed description of k-means clustering, see [99]. Depending on the underlying real-world application, it might not be possible to calculate geometric center for a subset of points. For therapist routing the modified Equation (6.37) is used:

$$\min \sum_{i=1}^{k} \sum_{j \in \mathscr{S}_i} \left(t_{l_j, l_j^{\text{center}}}\right)^2. \tag{6.37}$$

minimizing the travel time from the most centrally located location l_j^{center} to all other locations l_j in the cluster. Once the clusters have been generated, clusters are randomly selected and all customers in the selected clusters are removed until the desired number of removals has been performed. The number of cluster k can be set arbitrarily.

Zone destroy Similar to the simplified Shaw destroy, the zone destroy operator randomly selects one customer i with his/her location l_i. It then removes all customers, who *could* be assigned to one location within a given distance around location l_i. If the number of removed customers is below n, the distance around l_i is increased until n customers have been removed. Thereby, the operator does not only consider customers who are already close to one another but also customers who are currently served in another location but could also be served in the zone.

Subroute destroy A customer-location tuple is randomly selected and then, starting from this tuple, a virtual route of length n is constructed in a greedy fashion. Afterwards, all tuples in this virtual route are removed from the existing routes in the temporary solution.

Start time flexibility destroy In the hospital setting, customers have very different time window lengths. Outpatients generally have fixed appointments and, thus, a fixed start time, and some of the inpatiens have quite large time windows. The start time flexibility destroy first removes those customers who have the most flexibility in terms of possible start times. These customers are more likely to find another insertion position, while customers with fixed start times might already have a good position in the current solution.

Skill mismatch destroy As introduced in Section § 6.2.3.3, the model accounts for a heterogeneous fleet. In the hospital setting, the heterogeneity is expressed in the different skills of the therapists. Generally, a vehicle $k \in \mathscr{K}$ can serve customer $i \in \mathscr{I}$ in location $l \in \mathscr{L}$ if and only if vehicle k has the skill required for serving customer i. The skill mismatch destroy operator removes customers from a vehicle's route if the discrepancy between the required and the provided skill is largest. Let k^{skill} denote the provided skill level of vehicle k, and i^{skill} the required skill level of customer i, the skill discrepancy is defined as:

$$D^{\text{skill}}(i,k) = |k^{\text{skill}} - i^{\text{skill}}| \tag{6.38}$$

In order to enhance the exploration of the solution space, the following destroy operators are randomized: neighbor graph, simplified Shaw (proximity), time related, worst, skill mismatch, and time flexibility. This prevents that entities are removed in a deterministic way inhibiting a diverse exploration. The stochasticity is controlled by an operator-specific parameter $p^{(op)}$ where low values correspond to large randomness in the selection of a destroy operator.

6.2.4.3.4 Repair Operators

To reinsert the removed customers, three types of repair operators, namely greedy repair, κ-regret repair operators with κ ranging from 2 to 6, and skill match repair are employed. For the κ-regret repair, the same regret measure is used as in the construction heuristic (cf. Equation (6.28) in § 6.2.4.2).

Skill match Insertion The skill match insertion heuristic is based on the reverse idea of the equivalent destroy operator described in the previous section. This operator schedules customers $i \in \mathscr{I}$ to vehicles $k \in \mathscr{K}$ for which the discrepancy between the required and the provided skill level is small.

6.2.4.4 Implementation Details

During the execution of the algorithm, feasibility must be checked frequently. Testing for capacity violations is computationally expensive; however, it does not have to be done for all customer-location combinations. Because of the start time windows and the service duration, we know for some customers that serving them in a specific location will never lead to a capacity violation, since not enough other customers exist, who could be served in this location at this specific time. Therefore, to accelerate the algorithm, we determine in a preprocessing step all locations and corresponding time intervals which could have capacity violations. During the execution of the heuristic, the algorithm tests location capacity only for those customer-location combinations which could potentially lead to capacity violations.

6.2.5 Computational Study

In this section, the thesis investigates the performance of the proposed model. In particular, in § 6.2.5.1 it is described how the data used in the computational experiments has been generated. The thesis evaluates the newly introduced algorithm features in § 6.2.5.2. In particular, it is investigated the value of (a) the backtracking procedure in the construction phase, (b) the GLS, and (c) the newly introduced destroy operators. Furthermore, the heuristic is compared to current hospital planning in § 6.2.5.3. In addition to evaluating the solution quality, the thesis also studies the value of flexibility, i.e., how solutions change depending on different cost functions for customer travel times. Thereby, it is possible to trade off customer and vehicle travel times. Finally, in § 6.2.5.4, the performance of the algorithm for the VRPTW on the Solomon benchmark instances [194] is shown.

The algorithms were coded in JAVA using Amazon Corretto 11 as JDK and executed on an Ubuntu 20.04.2 LTS platform employing an AMD Ryzen Threadripper 1950X 16-Core Processor @ 3.40GhZ with 124 GB of RAM. The source code has been made publicly available at https://github.com/christianmaxmike/ALNS-VRPTW-FL.

6.2.5.1 Data and Instance Generation

Since no benchmark instances exist for the VRPTW-FL, instances based on data provided by a cooperating hospital were created. To account for different problem sizes and to ensure reliability of the results, we developed an instance generator to create generic problem instances. Three components define an instance: (a) the network layout representing the hospital, (b) the demand scenario representing customers (treatments), and (c) the fleet of vehicles representing therapists.[27]

Network layout This thesis distinguishes three layouts, which are defined by the number of buildings $B \in \{1,2,6\}$ and the number of floors per building $F \in \{1,3,6\}$. Each floor has a certain number of rooms (locations) $R \in \{6,7,\ldots,10\}$ drawn from a discrete uniform distribution. One of the buildings contains the therapy centers, which has a capacity between 2 and 6. The capacity at the ward rooms is always unlimited since patients cannot be scheduled to other patients' ward room. The travel time between two buildings is drawn at random from the set $\{10,15,20\}$ minutes. The travel time between neighboring floors is assumed to be 5 minutes, and the travel time between two rooms on the same floor is either 5 or 10 minutes. As stated above, travel costs are equated with travel times.

Demand scenarios The thesis distinguishes six demand scenarios having 20, 40, 60, 80, 100 and 120 treatments. A 10% probability exists that the patient is an *outpatient*, i.e., he/she can only be treated in a therapy center and the start time for the treatment is fixed. Ten percent of the patients are *bedridden*, i.e., the patient must not be moved and can only be treated in his/her room

[27] Data publicly available: https://zenodo.org/record/6772201#.YrsDJJDP3Kp

at the ward. However, these latter patients have a rather wide start time window of 90 minutes. The remaining patients are *regular inpatients*, of which 50% have location flexibility, i.e., the patient can be treated at the ward and in the therapy center. For each patient with location flexibility, the preferred room is selected randomly with each room having the same selection probability. For each preferred location the location costs is set to 0 and for the non-preferred locations the location costs are set to the distance from the preferred location. By this, the additional effort for transporting the patient between locations are reflected. In § 6.2.5.3.2, the thesis reports about a computational study where the location costs were systematically varied. The start time window length of these patients varies between 30 and 45 minutes. Thirty percent of the patients receive multiple treatments (2 or 3) in one day.

Every treatment job has a duration of 10 to 45 minutes and requires a certain skill level. Let us assume hierarchical skills ranging from 1 (lowest) to 3 (highest). The probabilities that a job requires a certain skill are 60%, 30% and 10% for skills 1, 2 and 3, respectively.

Vehicles A heterogeneous fleet is used, since therapists differ in their skills as well as their shift patterns. The skills are the same as for the jobs; however, the probabilities of having skill 1, 2 and 3 are 10%, 60% and 30%. A therapist has a regular (long) shift with 80% probability. Otherwise, the therapist has a short shift with 50% probability of being a morning or evening shift. We assume that therapists start and end their shifts in the break room (depot), which is 5 minutes away from the therapy centers.

Final instance set Two sets of data were generated: a training set and a test set. The training set is used to pre-test the features of the heuristic and to tune its parameters, and the test set is used for the numeric study. Each set consists of $5 \cdot 3 \cdot 6 = 90$ instances, as five instances were created for each combination of the three layouts and six demand scenarios. As the daily scheduling problem decomposes into a morning and afternoon part separated by a lunch break, and following Jungwirth et al. [103], the instances were split into a morning and an afternoon part each containing approximately half the number of customers.

6.2.5.2 Evaluation of Algorithmic Features

The thesis introduced three essential features for the ALNS: backtracking in the construction heuristic, a GLS to penalize violations of constraints, and new destroy operators specifically tailored for a problem structure, where multiple service locations exist for customers. For each feature, it first evaluates the benefit of the individual features by comparing the performance of the heuristic with and without the feature, and then it evaluates the performance of all features combined.

6.2.5.2.1 Value of Backtracking

The backtracking mechanism adds a look-ahead perspective to the construction heuristic to alter unsatisfactory decisions during the insertion process (cf. § 6.2.4.2). To evaluate the value of

backtracking, the construction heuristic including backtracking is compared to a version without backtracking, i.e., the best solution generated by the greedy and κ-regret insertions with $\kappa = \{2, 3, \ldots, 6\}$. For backtracking, stepping back to the first five inserted customers is allowed with the following probabilities 1.0, 0.6, 0.3, 0.2 and 0.1, i.e., a 10% probability exists to step back to the fifth customer, and if this is rejected, the procedure steps back to the fourth customer with a probability of 20%, etc. These probabilities are independent of one another, i.e., for each stage to which it can be backtracked, a new random number is drawn if backtracking was rejected in the prior stages.

The result of the comparison of the construction heuristic with and without backtracking can be found in Table 6.1. For each combination of layout (B, F) and demand scenario ($|\mathscr{I}|$), it is displayed the average values over five instances and three runs for the objective function value $f(s^*)$ of the best solution found s^*, the number of not assigned customers in that solution $|I^{\text{na}}(s^*)|$, the number of precedence violations $|I^{\text{pred}}(s^*)|$, the percentage of feasible solutions n^{feas}, and the number of backtrackings n^{bt}. Note that neither time window nor skill violations are possible during the construction phase.

By enabling backtracking, the solution quality after the construction phase could be increased substantially. On average, the objective function value was improved by 30.70% , the numbers of not assigned customers by 91.22%, and precedence violations were completely eliminated. The fraction of instances for which feasible solutions were found increased by 45.5% up to 89%. The number of backtrackings used per instance averaged at 1076.36, where in the evaluation, the number of maximal backtracking steps was set to 5000.

A proper initialization is critical for the ultimate model's performance as an inappropriate initial solution might lead to a subpar local minimum and, therefore, leads to a slow-down in the convergence of the algorithm. A good initial solution comes with the cost of larger running times for the construction phase. Due to the exponential growth of the tree being explored during the backtracking, a full depth-first search is too time-consuming. Therefore, the number of backtrackings are restricted and a termination criteria for the backtracking heuristic is fulfilled whenever the maximal number of backtracking is reached.

Table 6.1: *Comparison of construction heuristic with and without backtracking. For each variant, the objective function value $f(s^*)$ (Equation (6.27)), the number of not assigned customers $|I^{na}(s^*)|$, the number of precedence violations $|I^{pred}(s^*)|$, the percentage of feasible solutions n^{feas}, and the run-time t in [s] is displayed. For the backtracking variant, additionally the number of backtrackings n^{bt} is displayed. Each row represents the average values of all five instances per setting, each ran three times.*

$\|\mathscr{I}\|$	B	F	without backtracking					with backtracking					
			$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	n^{feas}	t	$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	n^{feas}	t	n^{bt}
20	1	6	52.00	0.1		0.9	0.02	46.7			1.0	0.06	12.1
	2	3	63.00	0.4		0.6	0.02	45.7	0.1		0.9	0.06	757.8
	6	1	53.60	0.3		0.7	0.02	44.2			1.0	0.04	10.6
40	1	6	136.10	1.4		0.2	0.03	91.5	0.4		0.6	0.76	2033.4
	2	3	137.30	1.4		0.2	0.03	84.1	0.3		0.7	2.37	1905.0
	6	1	114.80	1.6		0.3	0.03	80.4	0.4		0.7	1.46	2042.9
60	1	6	191.30	2.2		0.2	0.04	123.2	0.3		0.7	7.60	2581.3
	2	3	179.87	1.5	0.1	0.2	0.05	110.6			1.0	0.89	416.7
	6	1	157.00	1.7		0.3	0.04	120.5	0.5		0.6	2.14	2046.1
80	1	6	226.64	2.2	0.2	0.1	0.08	147.6	0.2		0.8	15.92	1599.1
	2	3	246.50	2.2		0.1	0.07	151.3			1.0	8.21	625.8
	6	1	210.44	2.3	0.2	0.2	0.06	148.6	0.1		0.9	10.91	875.3
100	1	6	265.67	2.0	0.1	0.2	0.11	196.9	0.1		0.9	15.94	811.9
	2	3	274.60	2.2			0.11	176.3			1.0	6.87	129.5
	6	1	246.70	2.7			0.14	174.8			1.0	9.48	714.8
120	1	6	333.54	2.7	0.2	0.2	0.18	241.8	0.3		0.8	59.02	1079.9
	2	3	315.27	2.1	0.1	0.1	0.21	217.7			1.0	48.50	911.3
	6	1	298.74	2.9	0.2	0.1	0.23	225.7	0.1		0.9	56.95	821.0
Avg.			194.61	1.77	0.06	0.26	0.08	134.87	0.16	0.00	0.86	13.73	1076.36
Δ								-30.70%	-91.22%	-100.00%	+236.96%	+16715%	

6.2.5.2.2 Value of ALNS+GLS

A standard ALNS working on objective function (6.27) was tested against an ALNS working on objective function (6.25), which dynamically adjusts penalizing infeasibilities. Since both version use different objective functions, the development of the best feasible solution is used as the metric for fair comparison. Figure 6.6 shows the development for all runs on a logarithmic scale. For each iteration the gap to best known solution states how far the best feasible solution of the current run is apart from the best feasible solution over all runs for the same instance.

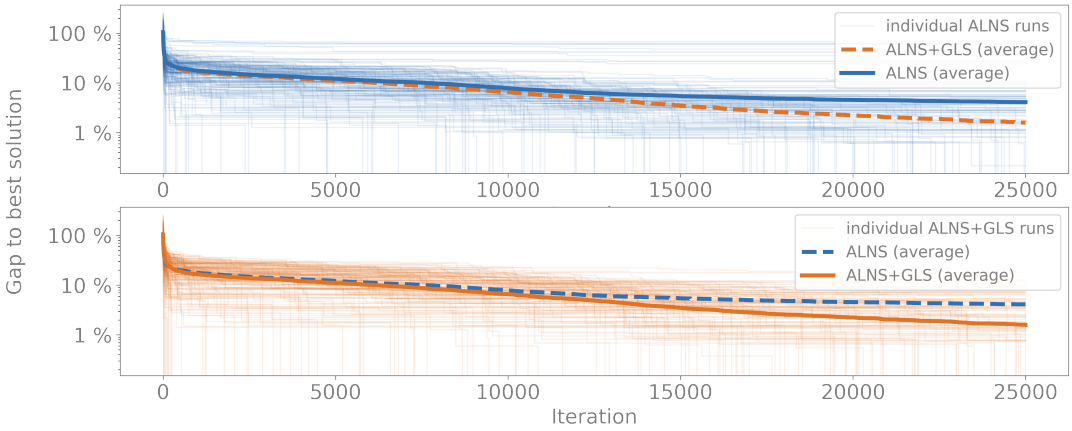


Figure 6.6: *Comparison of ALNS (above, blue lines) and ALNS+GLS (below, orange lines). Thin lines represent the percentage gap to the best feasible solution found over all runs. The bold lines represent the average percentage gap.*

The ALNS with GLS is able to get into much better regions of the solution space and is able to improve feasible solutions even in later iterations.

Table 6.2 shows the percentage gaps after n iterations. The gap between the two algorithms increases with increasing number of customers. The gap also increases slightly with the number of iterations, from 2.32% at $n = 10,000$ to 2.53% at $n = 25,000$.

While the average performances of the ALNS and ALNS+GLNS are sufficiently good for real-world hospital operations, Figure 6.6 shows a considerable variance in the performances over the entire instance set. Strong inter-dependencies between the parameters steering the ALNS and the parameters steering the GLS could be observed. An in-depth analysis of these inter-dependencies, and an instance-based parameter tuning that, e.g., considers different hospital layouts, is a fruitful area of further research. Automated design for meta- and matheuristics as described by Stützle and López-Ibáñez [197] and Maniezzo et al. [143] seems to be a promising starting point.

Table 6.2: *Achieved gap reduction in percent after n iteration by using ALNS+GLS compared to ALNS. Each row represents the average values of all five instances per setting, each ran three times.*

$\lvert\mathscr{I}\rvert$	B	F	$\Delta^{\text{gap}}_{n=5{,}000}$	$\Delta^{\text{gap}}_{n=10{,}000}$	$\Delta^{\text{gap}}_{n=15{,}000}$	$\Delta^{\text{gap}}_{n=20{,}000}$	$\Delta^{\text{gap}}_{n=25{,}000}$
20	1	6	3.19	2.69	2.69	2.69	2.69
	2	3	4.26	4.26	4.26	4.26	4.26
	6	1	-1.25	-1.35	-1.21	-1.21	-1.21
40	1	6	14.47	14.23	14.78	14.78	14.78
	2	3	17.79	18.90	18.90	18.90	18.90
	6	1	3.66	3.86	4.00	4.00	4.00
60	1	6	5.26	7.16	7.15	7.15	7.15
	2	3	-0.43	-1.94	-1.49	-1.49	-1.49
	6	1	2.33	4.83	5.41	5.41	5.41
80	1	6	-1.02	0.20	-0.34	-0.34	-0.34
	2	3	-0.83	1.54	1.49	1.49	1.49
	6	1	-1.54	-1.62	-1.15	-1.15	-1.15
100	1	6	-3.67	0.49	0.91	0.91	0.91
	2	3	-3.39	-2.98	-3.45	-3.45	-3.45
	6	1	-3.80	-1.41	-1.49	-1.49	-1.49
120	1	6	-1.65	-1.51	0.57	0.57	0.57
	2	3	-6.26	-3.19	-3.42	-3.42	-3.42
	6	1	-3.88	-2.47	-2.08	-2.08	-2.08
Avg.			1.29	2.32	2.53	2.53	2.53

6.2.5.2.3 Value of New Operators

To specifically tackle the underlying problem structure, several new neighborhoods (cf. § 6.2.4.3.3) were developed. The thesis investigates their impact by analyzing the probability of selecting a specific operator over time. The more successful a certain operator has been in earlier iterations, the more likely it will be selected in later iterations. The results for the destroy operators are given in Figure 6.7 and the results for the repair operators are given in Figure 6.8. The graphics are both organized in the same way. Each tile highlights the selection probability for one operator averaged over all instances and all runs.

Most of the operators generally used in the literature perform well and most of the newly developed operators can add additional value. Especially, the start time flexibility destroy and the skill mismatch destroy provide significant value, and k-means clustering is a valid alternative to clustering based on Kruskal's minimal spanning trees.

Noteworthy, the location-related and location-and-time-related destroy operators did not perform well. A possible explanation might be, that the location that is shared most by customers is the therapy center. The therapy center, however, is capacitated which limits the amount of customers, which can be placed in this neighborhood.

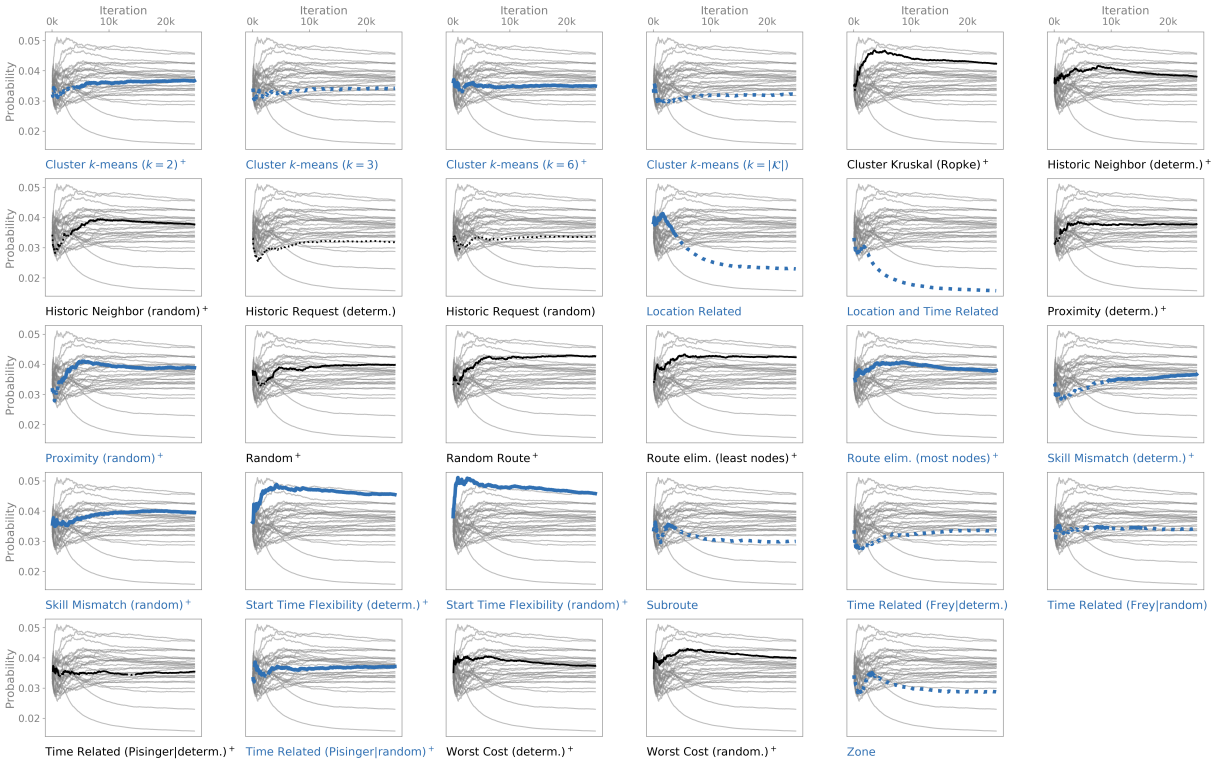


Figure 6.7: *Probabilities of drawing the destroy operators over* 25,000 *iterations. Each subplot displays the average probability of an operator over the* 90 *instances. Black thin lines represent operators from the literature. Blue thick lines represent new operators. Light gray lines represent the operators which are not the focus of the subplot. Above average performance is displayed by solid line segments while dotted lines represent under average performance.*

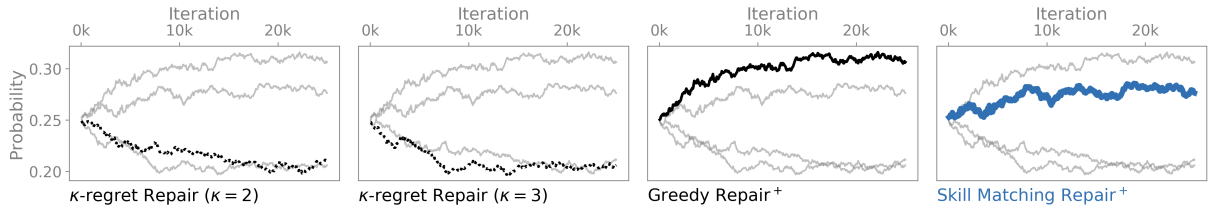


Figure 6.8: *Probabilities of drawing the repair operators over* 25,000 *iterations. Each subplot displays the average probability of an operator over the* 90 *instances. Black thin lines represent operators from the literature. Light gray lines represent the operators which are not the focus of the subplot. Above average performance is displayed by solid line segments while dotted lines represent under average performance.*

With the exception of the two location related destroys, all the averaged results are rather close together. The main reason is that the individual performance equals out over 180 instances repeated three times. However, performance in the individual runs varies significantly. To present a typical example, Figure 6.9 displays the distribution of the individual runs for the *Location-related destroy*.

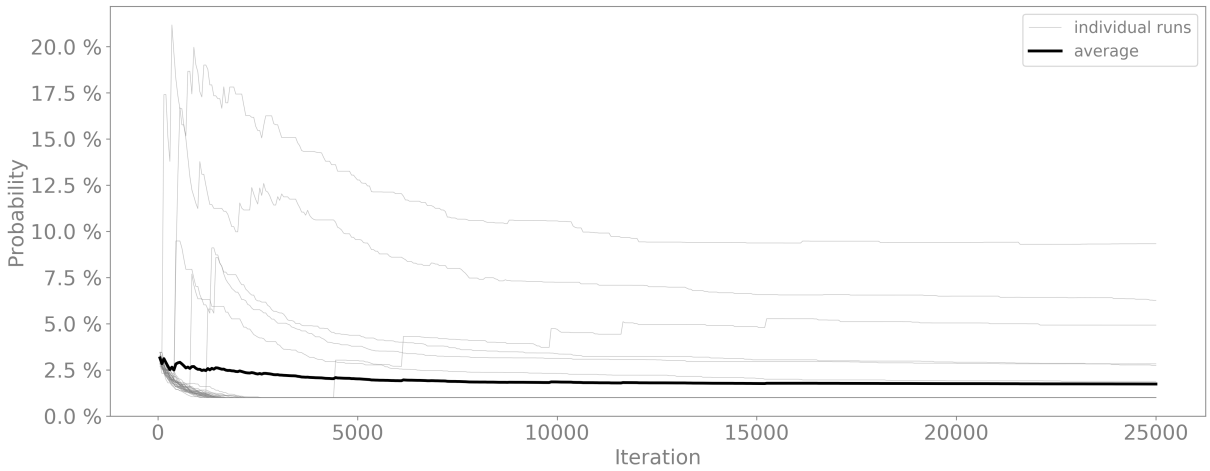


Figure 6.9: *Distribution of the probability developments over time for the location-related destroy. Thin gray lines represent the individual runs, while the bold black line represents the average over all runs.*

The solution qualities of the ALNS with and without the new operators are similar. However, for bigger instances (100, 120 customers) slightly better performance was observed when using the new operators. The number of unscheduled customers decreases by 6.67% to 0.078, the objective function value decreased imperceptibly from 95.64% to 95.14% (-0.52%), and the average runtime also decreased lightly from 67.0 seconds to 66.57 (-0.66%).

6.2.5.2.4 Value of All Features

After having tested the algorithmic features individually, it is important to examine how the features interact when used together. The results of testing a standard ALNS against one with backtracking in the construction, GLS and new operators are displayed in Table 6.3.

Moderate improvements are achieved when using all features. The objective function value is improved by 1.1% mainly due to consistently reducing the number of unscheduled customers (-80%). However, this comes with an increase in number of time window violations (12%) and skill violations (3%). Thus, the ratio of instances where the best solution found was feasible decreased to 85% (-7.3% over standard ALNS).

Table 6.3: *Comparison of basic ALNS without new features and ALNS with all new features (i.e., backtracking in construction, additional destroy operators, and GLS). For each variant, the objective function value $f(s^*)$ (Equation (6.27)), the number of not assigned customers $|I^{na}(s^*)|$, the number of time window violations $|I^{tw}(s^*)|$, the number of precedence violations $|I^{pred}(s^*)|$, the number of skill violations $|I^{skill}(s^*)|$, the percentage of feasible solutions n^{feas}, and the run-time t in [s] is displayed. Each row represents the average values of all five instances per setting, each ran three times.*

$\|\mathscr{I}\|$	B	F	basic ALNS							ALNS with all new features						
			$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{tw}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	$\|I^{\text{skill}}(s^*)\|$	n^{feas}	t	$f(s^*)$	$\|I^{\text{na}}(s^*)\|$	$\|I^{\text{tw}}(s^*)\|$	$\|I^{\text{pred}}(s^*)\|$	$\|I^{\text{skill}}(s^*)\|$	n^{feas}	t
20	1	6	37.8					1.0	3.07	36.84		0.1			0.9	4.20
	2	3	35.4					1.0	2.86	33.80					1.0	3.81
	6	1	35.6					1.0	3.10	35.14				0.1	0.9	4.48
40	1	6	75.2	0.4				0.6	8.18	65.98	0.1	0.4			0.5	14.29
	2	3	67.5	0.3				0.7	9.26	55.74		0.3			0.7	14.79
	6	1	61.1	0.3				0.7	8.31	58.52	0.2	0.2			0.6	13.55
60	1	6	88.4	0.2				0.8	21.93	83.02		0.3			0.8	42.54
	2	3	76.1					1.0	22.78	77.02		0.1		0.1	0.8	41.97
	6	1	88.5	0.2				0.8	23.16	83.78		0.3			0.8	43.39
80	1	6	102.1					1.0	53.24	101.60					1.0	113.06
	2	3	108.2					1.0	52.53	110.00					1.0	105.25
	6	1	103.9	0.1				0.9	55.88	106.38		0.1		0.1	0.8	110.58
100	1	6	139.0					1.0	99.86	137.68		0.1			0.9	222.31
	2	3	119.4					1.0	104.30	125.50					1.0	237.68
	6	1	120.2					1.0	107.18	122.74				0.1	0.9	206.39
120	1	6	160.9					1.0	197.50	160.96		0.2			0.8	389.07
	2	3	147.4					1.0	192.53	151.54				0.1	0.9	380.54
	6	1	154.9					1.0	240.50	156.10					1.0	408.37
Avg.			95.64	0.08	0.00	0.00	0.00	0.92	67.01	94.57	0.02	0.12	0.00	0.03	0.85	130.90
Δ										-1.1%	-80.0%				-7.3%	+95.4%

6.2.5.3 VRPTW-FL Applied to Therapist Scheduling and Routing

After investigating the algorithmic features in the last part, this part focuses on the hospital case and the value an ALNS can add to current planning practice.

6.2.5.3.1 VRPTW-FL Compared to Manual Planning

To compare the proposed ALNS to current hospital planning, the sequential allocation heuristic (SAH) was used as described in Gartner et al. [70]. According to Gartner et al. [70], this approach resembles manual planning. The central idea of the SAH is to separate customers with fixed start times, which are assigned first, from those with flexible start times, which are assigned later. Sorting is used to facilitate assignment of customers to good tours, e.g., vehicles are sorted in order of increasing shift start times and customers are sorted by earliest start time.

The results for the SAH and the proposed ALNS are given in Table 6.4. The novel approach presented in this thesis clearly outperforms the SAH by increasing the ratio of feasible solutions found from 29% up to 85%. The ALNS leaves almost no customers unscheduled and reduces the number of precedence violations by 100%. Reducing precedence violations is highly relevant for practice because in most cases, medical reasons exist for those precedence relations. However, this comes at the cost of violating the desired time windows in 11% of the instances and the violating the required skill level in 3% of the instances.

6.2.5.3.2 Impact of Location Costs

In order to assess the impact of different location costs, four different cost functions are considered for assigning a customer to a non-preferred location: (1) no costs, (2) costs of one, (3) costs equal to the distance between preferred location and assigned location $t_{r,l}$, and (4) costs equal to this distance squared $(t_{r,l})^2$.

Table 6.5 summarizes the results for the different cost functions. For increasing location costs, the share of patients served at the preferred location becomes greater. At the same time, the total distance traveled by the therapists increases. When comparing the extreme cases, location costs of 0 and $(t_{r,l})^2$, the average ratio of patients treated in the preferred location increases by 255% from 37.8% to 96.2%, and the travel distance of the therapists increase by 224% from 35.7 to 80.02.

Table 6.4: *Comparison of hospital planning and ALNS. For each variant, the objective function value $f(s^*)$ (Equation (6.27)), the number of not assigned customers $|I^{na}(s^*)|$, the number of time window violations $|I^{tw}(s^*)|$, the number of precedence violations $|I^{pred}(s^*)|$, the number of skill violations $|I^{skill}(s^*)|$, and the percentage of feasible solutions n^{feas} is displayed. Each row represents the average values of all five instances per setting, each ran three times.*

$\|\mathscr{I}\|$	B	F	hospital planning						ALNS+GLS					
			$f(s^*)$	$\|I_i^{\text{na}}(s^*)\|$	$\|I_i^{\text{tw}}(s^*)\|$	$\|I_i^{\text{pred}}(s^*)\|$	$\|I^{\text{skill}}(s^*)\|$	n^{feas}	$f(s^*)$	$\|I_i^{\text{na}}(s^*)\|$	$\|I_i^{\text{tw}}(s^*)\|$	$\|I_i^{\text{pred}}(s^*)\|$	$\|I^{\text{skill}}(s^*)\|$	n^{feas}
20	1	6	51.6	0.1				0.8	36.84		0.1			0.9
	2	3	49.5					1.0	33.80					1.0
	6	1	46.3					1.0	35.14				0.1	0.9
40	1	6	100.3	1.1					65.98	0.1	0.4			0.5
	2	3	89.9	0.6		0.2			55.74		0.3			0.7
	6	1	87.6	0.4		0.4		0.2	58.52	0.2	0.2			0.6
60	1	6	140.2	1.0		0.6			83.02		0.3			0.8
	2	3	138.2	0.1		0.4		0.6	77.02		0.1		0.1	0.8
	6	1	139.2	1.3		0.4			83.78		0.3			0.8
80	1	6	169	0.6		0.2		0.2	101.60					1.0
	2	3	185.3	1.0		0.2		0.4	110.00					1.0
	6	1	177.9	1.2		0.2		0.2	106.38		0.1		0.1	0.8
100	1	6	242.1	1.5		0.8			137.68		0.1			0.9
	2	3	220.9	0.8		0.4		0.4	125.50					1.0
	6	1	214.9	0.5		1.0		0.2	122.74				0.1	0.9
120	1	6	280.1	0.9		0.6			160.96		0.2			0.8
	2	3	263.2	0.9		0.4		0.2	151.54				0.1	0.9
	6	1	260.6	1.6		0.6			156.10					1.0
Avg.			158.71	0.76	0.00	0.35	0.00	0.29	95.57	0.02	0.11	0.00	0.03	0.85
Δ									-40.1%	-97.8%		-100.0%		+194.2%

Table 6.5: *Impact of location costs. Each row represents the average values of all five instances per setting, each ran three times.*

$\lvert\mathscr{I}\rvert$	B	F	distance therapists				frac. preferred locations			
			0	1	$t_{r,l}$	$(t_{r,l})^2$	0	1	$t_{r,l}$	$(t_{r,l})^2$
20	1	6	14.30	14.44	27.86	32.90	0.29	0.34	0.82	0.96
	2	3	14.80	14.80	23.50	34.64	0.37	0.45	0.71	0.93
	6	1	16.16	16.43	23.50	34.20	0.40	0.44	0.71	0.98
40	1	6	35.36	37.24	47.80	56.64	0.53	0.62	0.85	0.94
	2	3	32.26	32.66	43.34	53.04	0.60	0.64	0.84	0.96
	6	1	35.70	36.40	43.24	51.46	0.65	0.71	0.86	0.97
60	1	6	37.75	38.54	61.54	69.83	0.48	0.56	0.87	0.94
	2	3	30.40	31.64	56.84	68.40	0.36	0.48	0.86	0.97
	6	1	38.76	39.74	57.74	77.50	0.46	0.53	0.80	0.97
80	1	6	36.76	39.10	74.76	87.76	0.34	0.47	0.85	0.94
	2	3	40.03	41.66	81.54	93.56	0.32	0.42	0.89	0.97
	6	1	40.40	42.56	69.44	89.24	0.34	0.45	0.83	0.98
100	1	6	46.46	48.66	96.50	113.06	0.30	0.41	0.88	0.97
	2	3	41.24	43.90	81.80	100.56	0.30	0.46	0.85	0.97
	6	1	42.58	44.42	78.54	102.75	0.30	0.42	0.81	0.97
120	1	6	50.84	53.10	109.94	129.40	0.27	0.37	0.86	0.96
	2	3	43.70	43.90	94.36	123.10	0.27	0.38	0.82	0.97
	6	1	45.04	47.60	96.54	122.34	0.24	0.36	0.82	0.96

Figure 6.10 displays this causal relationship grouped by the different demand scenarios (number of customers). When costs are $(t_{r,l})^2$, deviating from the preferred locations is avoided whenever possible, only in cases of limited capacity in the therapy center an alternative location is assigned. Increasing the cost for alternative locations from 0 to 1 does only have a very small impact on the travel costs of therapists. For costs of 1, location preferences are almost entirely neglected, i.e., patients will be scheduled to alternative rooms although this only slightly improves the travel distances of the therapists. Presumably, a good trade-off in hospital planning is to use location costs equivalent to the distance between the preferred location and the assigned locations. However, this might not generalize to other routing contexts with very different underlying network structures.

6.2.5.4 ALNS on Solomon Instances

To show the general capabilities of the novel algorithm, its performance for the VRPTW on the well-known Solomon benchmark instances [194] were evaluated. Note that the algorithm's parameters were not tuned for the Solomon instances, instead the parameters obtained by tuning for the hospital instances were used as described in § 6.2.5.1. The results for the 29 instances

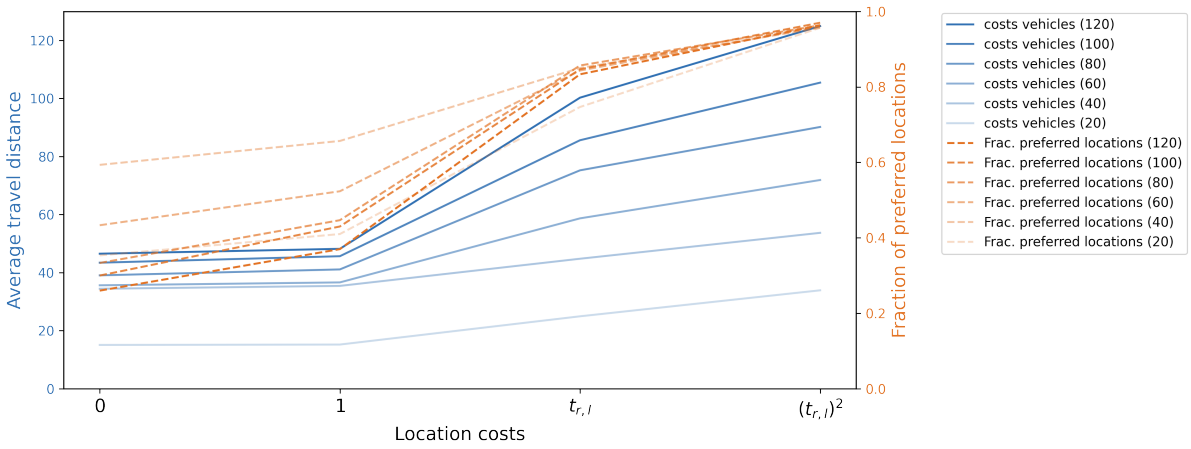


Figure 6.10: *Travel distances of therapists (vehicles) in blue and fraction of treatments in preferred locations in orange for different location costs* $(0,\ 1,\ t_{r,l},\ (t_{r,l})^2)$*. Each line groups the results for different instances sizes* $(20,\ 40,\ \ldots,\ 120$ *customers).*

of the first class with relatively narrow time windows are given in Table 6.6. Considering that the proposed algorithm is not tuned for these instances, it performs well. For the 25-customer instances, optimality is reached in all cases, for 50 customer in 7, and for 100 customers in 2 cases. The optimality gaps are larger than, e.g., in Vidal et al. [208], which is expected as the proposed algorithm was developed to cope with the very distinct properties of the VRPTW-FL.

Table 6.6: *Performance of the ALNS on Solomon benchmark instances. The types are clustered (C), random (R), partially random/clustered (RC), optimum is the optimal solution, ALNS is the average result achived by the ALNS over five runs, gap is the relative gap between ALNS and the optimal solution, and n optimal states how often the optimal solution was found for each instance.*

type	customers	optimum	ALNS	gap	*n* optimal
C	25	190.59	190.59	0.0	9 / 9
	50	361.69	373.83	0.03	5 / 9
	100	826.70	944.36	0.14	1 / 9
R	25	463.37	463.37	0.0	12 / 12
	50	766.13	777.02	0.02	2 / 12
	100	1173.61	1258.68	0.08	0 / 12
RC	25	350.24	350.78	0.0	6 / 8
	50	730.31	736.14	0.01	0 / 8
	100	1334.49	1444.39	0.09	0 / 8

6.3 Summary

The VRPTW-FL is a relevant and highly complex problem. Considering multiple possible service locations in routing problems receives increasing interest in the VRP community and this thesis addressees a mathematical formulation of location capacities in the service locations for the first time. The VRPTW-FL occurs in scheduling physical therapists for which the solution approach was developed. The proposed model built on an ALNS framework and was enhanced with several innovative ideas, i.e., (a) a backtracking procedure in the construction phase to correct poor assignment of customers to vehicles, (b) a guided local search (GLS) that dynamically adjusts how infeasibilities are penalized, and (c) new neighborhoods exploiting the underlying problem structure.

The algorithm was evaluated on a set of generic instances designed to represent different hospital layouts and different demand scenarios. The computation results show that the developed enhancements add value to better solving the VRPTW-FL. The thesis generated insights how different cost functions for assigning customers to location different from the preferred location affect the overall planning. These insights can be used by hospital managers to decide which cost function to be used depending on their healthcare system and/or customer preferences. A more in-depth investigation of the interplay between hyperparameters steering the ALNS and the parameters steering the GLS is likely to lead to a more robust version of the novel algorithm and presents a promising research direction.

The authors of the joint work [66] – which has been presented in this chapter – believe that VRPTW-FL and related problems will receive more attention in near future. Savelsbergh and Woensel [180] describe these problems as an opportunity in city logistics and we believe that for many applications location capacities become a limiting factor, e.g., limited parking space availabilities are so far completely ignored in the OR literature.

6.4 Addendum

6.4.1 Detailed Computational Results

Table 6.7 lists the computational details for the individual instances. The instance name *name* is composed of *i* followed by the number of customers in the instance, *b* followed by number of buildings and *f* followed by the number of floors per building defining the hospital layout, *v* the instance number for this combination of customers and hospital layout, and either *m* for the morning part or *a* for the afternoon part. Further, the following notation is used: number of customers in the morning/afternoon part $|\mathscr{I}|$, fitness value of the solution after the construction phase s^{c}, number of backtracking steps during the construction phase n^{bt}, computation time for the construction phase t^{c}, fitness value after the ALNS phase s^{alns}, computation time for the ALNS pahse t^{alns}, number of vehicles used $\mathscr{K}^{\mathrm{used}}$, binary indicator if the solution is feasible *feas*, number of non-assigned customers n^{na}, number of timewindow violations n^{tw}, number of precedence violations n^{pred}, and number of skill violations n^{skill}.

Table 6.7: Detailed results for ALNS with all new features (i.e., backtracking in construction, additional repair and destroy operators, and GLS)

id	name	$\lvert\mathscr{I}\rvert$	s^{c}	n^{bt}	t^{c}	s^{alns}	t^{alns}	$\mathscr{K}^{\mathrm{used}}$	feas.	n^{na}	n^{tw}	n^{pred}	n^{skill}
1	i020-b1-f6-v1-m	10	52.0		0.04	37.0	5.36	3	1				
2	i020-b1-f6-v1-a	10	37.0		0.01	35.0	4.12	3	1				
3	i020-b1-f6-v2-m	8	42.0		0.04	29.0	2.72	2	1				
4	i020-b1-f6-v2-a	12	51.0	2	0.03	42.0	5.45	3	1				
5	i020-b1-f6-v3-m	7	36.0		0.04	26.0	2.82	2	1				
6	i020-b1-f6-v3-a	13	51.0	1	0.03	40.0	6.32	4	1				
7	i020-b1-f6-v4-m	12	53.0	118	0.32	43.4	5.41	3			1		
8	i020-b1-f6-v4-a	8	50.0		0.01	40.0	2.19	3	1				
9	i020-b1-f6-v5-m	11	58.0		0.04	42.0	4.11	3	1				
10	i020-b1-f6-v5-a	9	37.0		0.02	34.0	3.51	3	1				
11	i020-b2-f3-v1-m	10	53.0		0.05	41.0	3.84	3	1				
12	i020-b2-f3-v1-a	10	53.0		0.02	35.0	4.08	3	1				
13	i020-b2-f3-v2-m	13	35.0	7	0.09	29.0	5.64	3	1				
14	i020-b2-f3-v2-a	7	30.0		0.01	28.0	2.33	2	1				
15	i020-b2-f3-v3-m	8	33.0		0.05	26.0	3.60	3	1				
16	i020-b2-f3-v3-a	12	49.0	2	0.03	48.0	5.00	3	1				
17	i020-b2-f3-v4-m	11	54.0	2501	0.10	37.0	4.24	4	1				
18	i020-b2-f3-v4-a	9	40.0	3	0.01	31.0	2.46	3	1				
19	i020-b2-f3-v5-m	11	39.0	65	0.16	31.0	4.18	3	1				
20	i020-b2-f3-v5-a	9	71.0	5000	0.06	32.0	2.75	3	1				
21	i020-b6-f1-v1-m	13	63.0		0.06	49.4	5.35	4					1
22	i020-b6-f1-v1-a	7	44.0		0.01	37.0	2.56	4	1				
23	i020-b6-f1-v2-m	9	41.0	6	0.05	31.0	3.70	3	1				

Table 6.7: Detailed results for ALNS with all new features (i.e., backtracking in construction, additional repair and destroy operators, and GLS) (continued)

id	name	$\lvert\mathscr{I}\rvert$	s^{c}	n^{bt}	t^{c}	s^{alns}	t^{alns}	$\mathscr{K}^{used}$	feas.	n^{na}	n^{tw}	n^{pred}	n^{skill}
24	i020-b6-f1-v2-a	11	41.0		0.02	28.0	3.71	3	1				
25	i020-b6-f1-v3-m	12	55.0		0.05	46.0	6.91	4	1				
26	i020-b6-f1-v3-a	8	33.0	7	0.02	27.0	2.52	3	1				
27	i020-b6-f1-v4-m	15	58.0		0.06	45.0	10.16	3	1				
28	i020-b6-f1-v4-a	5	17.0	93	0.03	17.0	1.74	2	1				
29	i020-b6-f1-v5-m	10	42.0		0.06	35.0	4.22	3	1				
30	i020-b6-f1-v5-a	10	48.0		0.03	36.0	3.92	3	1				
31	i040-b1-f6-v1-m	17	74.0	58	0.33	62.8	10.13	5			1		
32	i040-b1-f6-v1-a	23	116.0	5000	0.45	75.4	24.22	5			1		
33	i040-b1-f6-v2-m	22	87.0	195	0.84	64.0	16.18	5	1				
34	i040-b1-f6-v2-a	18	64.0	4	0.06	54.0	12.23	4	1				
35	i040-b1-f6-v3-m	19	94.0		0.09	75.0	14.54	7	1				
36	i040-b1-f6-v3-a	21	104.0	5000	2.76	93.0	12.76	4		1			
37	i040-b1-f6-v4-m	21	89.0	77	0.42	64.0	13.25	6	1				
38	i040-b1-f6-v4-a	19	80.0		0.09	50.0	11.72	5	1				
39	i040-b1-f6-v5-m	21	102.0	5000	1.61	54.8	13.15	4			1		
40	i040-b1-f6-v5-a	19	105.0	5000	0.90	66.8	14.72	6			1		
41	i040-b2-f3-v1-m	23	63.0	2146	5.20	54.0	17.62	5	1				
42	i040-b2-f3-v1-a	17	64.0	868	1.35	44.0	9.23	5	1				
43	i040-b2-f3-v2-m	17	93.0	5000	2.07	49.8	9.63	6			1		
44	i040-b2-f3-v2-a	23	83.0	13	0.12	58.0	18.17	5	1				
45	i040-b2-f3-v3-m	19	86.0		0.08	56.0	12.64	5	1				
46	i040-b2-f3-v3-a	21	129.0	5000	1.24	72.8	12.10	5			1		
47	i040-b2-f3-v4-m	16	65.0	3	0.09	50.0	8.95	4	1				
48	i040-b2-f3-v4-a	24	74.0	1003	4.49	58.0	32.54	5	1				
49	i040-b2-f3-v5-m	18	76.0	17	0.19	53.0	10.98	4	1				
50	i040-b2-f3-v5-a	22	108.0	5000	8.90	61.8	16.07	5			1		
51	i040-b6-f1-v1-m	24	102.0	2665	4.85	82.4	20.02	6			1		
52	i040-b6-f1-v1-a	16	76.0	5000	1.76	72.0	7.31	4		1			
53	i040-b6-f1-v2-m	24	85.0	2518	1.08	57.0	18.66	6	1				
54	i040-b6-f1-v2-a	16	47.0	228	0.74	35.0	9.27	4	1				
55	i040-b6-f1-v3-m	22	86.0	18	0.26	60.0	16.10	5	1				
56	i040-b6-f1-v3-a	18	77.0		0.04	57.0	13.26	4	1				
57	i040-b6-f1-v4-m	23	95.0	5000	4.07	58.8	16.63	6			1		
58	i040-b6-f1-v4-a	17	65.0		0.02	49.0	9.52	4	1				
59	i040-b6-f1-v5-m	23	108.0	5000	1.73	68.0	16.30	5		1			
60	i040-b6-f1-v5-a	17	63.0		0.01	46.0	8.39	4	1				
61	i060-b1-f6-v1-m	31	129.0	1682	4.92	90.0	42.49	7	1				
62	i060-b1-f6-v1-a	29	127.0	2555	3.77	90.0	51.65	7	1				
63	i060-b1-f6-v2-m	31	149.0	5000	4.64	91.0	40.48	7	1				
64	i060-b1-f6-v2-a	29	103.0	5	0.07	81.0	39.03	6	1				

Table 6.7: Detailed results for ALNS with all new features (i.e., backtracking in construction, additional repair and destroy operators, and GLS) (continued)

id	name	$\lvert\mathscr{I}\rvert$	s^{c}	n^{bt}	t^{c}	s^{alns}	t^{alns}	$\mathscr{K}^{used}$	feas.	n^{na}	n^{tw}	n^{pred}	n^{skill}
65	i060-b1-f6-v3-m	23	81.0		0.11	66.0	20.23	5	1				
66	i060-b1-f6-v3-a	37	153.0	5000	50.76	109.6	72.92	8			2		
67	i060-b1-f6-v4-m	31	120.0	195	1.76	88.0	37.43	7	1				
68	i060-b1-f6-v4-a	29	126.0	3791	2.38	81.0	50.22	6	1				
69	i060-b1-f6-v5-m	29	116.0	5000	4.05	63.6	34.09	6			1		
70	i060-b1-f6-v5-a	31	128.0	2585	3.56	70.0	36.88	6	1				
71	i060-b2-f3-v1-m	27	109.0		0.15	74.0	33.49	7	1				
72	i060-b2-f3-v1-a	33	111.0	15	0.21	70.4	47.44	7					1
73	i060-b2-f3-v2-m	27	104.0	7	0.27	71.0	33.54	6	1				
74	i060-b2-f3-v2-a	33	141.0	62	0.60	95.0	47.83	6	1				
75	i060-b2-f3-v3-m	29	98.0	25	0.72	75.0	40.22	7	1				
76	i060-b2-f3-v3-a	31	111.0	145	1.28	80.8	38.28	7			1		
77	i060-b2-f3-v4-m	27	118.0	19	0.38	88.0	31.32	6	1				
78	i060-b2-f3-v4-a	33	115.0	27	0.77	73.0	50.26	7	1				
79	i060-b2-f3-v5-m	30	99.0	16	0.39	74.0	39.32	7	1				
80	i060-b2-f3-v5-a	30	100.0	3851	4.16	69.0	57.98	6	1				
81	i060-b6-f1-v1-m	28	99.0	157	1.48	80.0	31.95	8	1				
82	i060-b6-f1-v1-a	32	140.0	5000	0.38	86.0	44.80	8	1				
83	i060-b6-f1-v2-m	34	168.0	5000	2.83	99.0	47.61	7	1				
84	i060-b6-f1-v2-a	26	114.0		0.04	82.0	23.80	8	1				
85	i060-b6-f1-v3-m	20	79.0		0.09	62.0	15.24	5	1				
86	i060-b6-f1-v3-a	40	139.0	5000	6.25	95.0	85.80	9			2		
87	i060-b6-f1-v4-m	38	127.0	291	4.72	94.0	79.82	8	1				
88	i060-b6-f1-v4-a	22	75.0	2	0.06	55.0	20.21	5	1				
89	i060-b6-f1-v5-m	33	144.0	5000	5.34	94.8	47.98	8			1		
90	i060-b6-f1-v5-a	27	120.0	11	0.19	90.0	36.68	6	1				
91	i080-b1-f6-v1-m	34	117.0	28	0.95	91.0	72.25	7	1				
92	i080-b1-f6-v1-a	46	158.0	2583	71.89	117.0	168.26	8	1				
93	i080-b1-f6-v2-m	49	157.0	2951	32.98	101.0	203.56	11	1				
94	i080-b1-f6-v2-a	31	122.0	43	0.56	91.0	53.09	6	1				
95	i080-b1-f6-v3-m	40	141.0	1	0.36	102.0	99.61	9	1				
96	i080-b1-f6-v3-a	40	158.0	31	1.10	120.0	102.19	10	1				
97	i080-b1-f6-v4-m	36	118.0		0.25	84.0	67.28	8	1				
98	i080-b1-f6-v4-a	44	151.0	354	7.95	110.0	118.61	9	1				
99	i080-b1-f6-v5-m	45	196.0	5000	32.28	113.0	114.07	9	1				
100	i080-b1-f6-v5-a	35	158.0	5000	10.86	87.0	131.65	8	1				
101	i080-b2-f3-v1-m	40	142.0	261	4.85	104.0	92.18	9	1				
102	i080-b2-f3-v1-a	40	126.0	497	12.35	98.0	141.98	8	1				
103	i080-b2-f3-v2-m	48	173.0	1748	29.30	110.0	162.79	10	1				
104	i080-b2-f3-v2-a	32	124.0	21	0.36	87.0	57.18	7	1				
105	i080-b2-f3-v3-m	37	148.0	136	2.45	110.0	73.55	9	1				

Table 6.7: Detailed results for ALNS with all new features (i.e., backtracking in construction, additional repair and destroy operators, and GLS) (continued)

id	name	$\lvert\mathcal{I}\rvert$	s^{c}	n^{bt}	t^{c}	s^{alns}	t^{alns}	$\mathcal{K}^{used}$	feas.	n^{na}	n^{tw}	n^{pred}	n^{skill}
106	i080-b2-f3-v3-a	43	159.0	121	2.81	130.0	110.34	9	1				
107	i080-b2-f3-v4-m	32	111.0	11	0.61	87.0	56.81	7	1				
108	i080-b2-f3-v4-a	48	191.0	2650	14.05	130.0	159.52	10	1				
109	i080-b2-f3-v5-m	42	170.0	813	15.27	124.0	111.57	11	1				
110	i080-b2-f3-v5-a	38	169.0		0.08	120.0	86.61	8	1				
111	i080-b6-f1-v1-m	34	119.0	6	0.36	90.0	57.67	9	1				
112	i080-b6-f1-v1-a	46	184.0	5000	59.01	125.4	139.06	9			1		
113	i080-b6-f1-v2-m	31	104.0	8	0.42	70.0	41.58	6	1				
114	i080-b6-f1-v2-a	49	186.0	2847	33.24	134.0	256.63	10	1				
115	i080-b6-f1-v3-m	44	145.0	233	5.40	103.0	131.77	9	1				
116	i080-b6-f1-v3-a	36	139.0	216	3.00	92.0	69.88	8	1				
117	i080-b6-f1-v4-m	39	158.0		0.29	104.0	99.43	8	1				
118	i080-b6-f1-v4-a	41	139.0	181	3.62	113.4	96.46	9					1
119	i080-b6-f1-v5-m	35	137.0	255	3.48	97.0	65.93	7	1				
120	i080-b6-f1-v5-a	45	175.0	7	0.31	135.0	147.43	9	1				
121	i100-b1-f6-v1-m	55	223.0	55	3.06	160.0	245.75	11	1				
122	i100-b1-f6-v1-a	45	175.0	11	0.45	137.8	128.75	10			1		
123	i100-b1-f6-v2-m	56	216.0	5000	53.09	138.0	244.59	11	1				
124	i100-b1-f6-v2-a	44	140.0	1396	12.13	110.0	148.59	9	1				
125	i100-b1-f6-v3-m	47	173.0	589	13.12	140.0	156.50	9	1				
126	i100-b1-f6-v3-a	53	212.0	940	70.19	144.0	450.93	10	1				
127	i100-b1-f6-v4-m	50	199.0	64	3.29	144.0	214.60	10	1				
128	i100-b1-f6-v4-a	50	192.0	4	0.50	137.0	234.52	10	1				
129	i100-b1-f6-v5-m	44	193.0		0.48	115.0	135.33	8	1				
130	i100-b1-f6-v5-a	56	246.0	60	3.07	151.0	263.59	11	1				
131	i100-b2-f3-v1-m	51	174.0	107	4.29	107.0	208.58	10	1				
132	i100-b2-f3-v1-a	49	168.0	45	1.64	128.0	207.98	10	1				
133	i100-b2-f3-v2-m	45	163.0	9	1.28	118.0	146.07	9	1				
134	i100-b2-f3-v2-a	55	187.0	208	20.95	126.0	571.09	11	1				
135	i100-b2-f3-v3-m	53	168.0	395	17.91	122.0	256.99	10	1				
136	i100-b2-f3-v3-a	47	148.0	32	1.30	114.0	171.12	11	1				
137	i100-b2-f3-v4-m	50	175.0	75	3.43	127.0	203.48	10	1				
138	i100-b2-f3-v4-a	50	147.0	189	7.41	112.0	202.93	9	1				
139	i100-b2-f3-v5-m	48	199.0	136	5.31	137.0	191.06	10	1				
140	i100-b2-f3-v5-a	52	234.0	99	5.15	164.0	217.45	10	1				
141	i100-b6-f1-v1-m	50	174.0	55	2.88	120.0	216.46	11	1				
142	i100-b6-f1-v1-a	50	163.0	3804	13.67	107.0	234.30	11	1				
143	i100-b6-f1-v2-m	56	220.0	44	3.25	156.4	262.93	14					1
144	i100-b6-f1-v2-a	44	157.0	165	5.00	102.0	146.85	10	1				
145	i100-b6-f1-v3-m	52	174.0	566	19.94	115.0	245.43	12	1				
146	i100-b6-f1-v3-a	48	172.0	18	1.19	128.0	197.31	9	1				

Table 6.7: Detailed results for ALNS with all new features (i.e., backtracking in construction, additional repair and destroy operators, and GLS) (continued)

id	name	$\lvert\mathscr{I}\rvert$	s^{c}	n^{bt}	t^{c}	s^{alns}	t^{alns}	$\mathscr{K}^{used}$	feas.	n^{na}	n^{tw}	n^{pred}	n^{skill}
147	i100-b6-f1-v4-m	49	149.0	1416	11.12	114.0	179.81	11	1				
148	i100-b6-f1-v4-a	51	162.0	160	7.16	116.0	199.81	9	1				
149	i100-b6-f1-v5-m	47	177.0	722	21.63	130.0	174.88	10	1				
150	i100-b6-f1-v5-a	53	200.0	198	8.99	139.0	206.13	11	1				
151	i120-b1-f6-v1-m	60	239.0	27	2.85	166.0	418.13	11	1				
152	i120-b1-f6-v1-a	60	242.0	84	7.83	164.0	376.96	12	1				
153	i120-b1-f6-v2-m	58	222.0	5000	207.58	150.0	322.66	11	1				
154	i120-b1-f6-v2-a	62	234.0	71	5.85	157.8	369.72	12			1		
155	i120-b1-f6-v3-m	48	187.0	4	0.92	135.0	225.98	10	1				
156	i120-b1-f6-v3-a	72	327.0	5000	310.03	201.8	718.60	16			1		
157	i120-b1-f6-v4-m	56	208.0	20	1.77	137.0	308.69	11	1				
158	i120-b1-f6-v4-a	64	258.0	544	48.75	166.0	410.90	12	1				
159	i120-b1-f6-v5-m	59	239.0	45	3.76	156.0	328.58	12	1				
160	i120-b1-f6-v5-a	61	262.0	4	0.89	176.0	410.50	12	1				
161	i120-b2-f3-v1-m	63	223.0	292	40.11	168.0	440.63	13	1				
162	i120-b2-f3-v1-a	57	265.0	25	1.84	167.0	334.91	12	1				
163	i120-b2-f3-v2-m	61	216.0	186	11.32	142.0	351.82	13	1				
164	i120-b2-f3-v2-a	59	180.0	3047	208.06	126.0	369.77	12	1				
165	i120-b2-f3-v3-m	65	206.0	9	1.68	155.0	443.73	13	1				
166	i120-b2-f3-v3-a	55	190.0	7	0.69	135.0	305.97	11	1				
167	i120-b2-f3-v4-m	56	212.0	2453	95.86	157.0	306.54	11	1				
168	i120-b2-f3-v4-a	64	245.0	2714	96.34	171.0	463.33	13	1				
169	i120-b2-f3-v5-m	61	221.0	39	3.92	156.4	391.98	11					1
170	i120-b2-f3-v5-a	59	219.0	341	25.18	138.0	396.75	11	1				
171	i120-b6-f1-v1-m	55	198.0	52	4.13	147.0	319.41	12	1				
172	i120-b6-f1-v1-a	65	234.0	1794	157.85	161.0	537.17	13	1				
173	i120-b6-f1-v2-m	72	270.0	5000	300.97	166.0	681.53	13	1				
174	i120-b6-f1-v2-a	48	171.0	10	0.60	130.0	206.55	11	1				
175	i120-b6-f1-v3-m	73	259.0	547	50.04	168.0	607.88	14	1				
176	i120-b6-f1-v3-a	47	181.0	6	0.54	121.0	188.12	10	1				
177	i120-b6-f1-v4-m	54	207.0	27	2.47	150.0	279.39	12	1				
178	i120-b6-f1-v4-a	66	253.0	103	9.77	175.0	470.97	13	1				
179	i120-b6-f1-v5-m	63	262.0	566	34.98	185.0	398.25	13	1				
180	i120-b6-f1-v5-a	57	222.0	105	8.20	158.0	394.42	11	1				

6.4.2 Overview ALNS+GLS Parameters

An overview of all hyperparameters used for the ALNS+GLS is provided in Table 6.8.

Table 6.8: ALNS parameters in alphabetical order

$\Delta = 0.05$	Deterioration parameters of initial solution; used to calculate start temperature
$\min(\rho) = 0.01$	Minimum probability of drawing a destroy or repair operator.
$\lambda^{\text{na}} = 12.8$	Weight for not assigning a customer
$\lambda^{\text{pred}} = 3.6$	Weight for violating a precedence relation
$\lambda^{\text{skill}} = 1.4$	Weight for violating a skill level by one time unit
$\lambda^{\text{tw}} = 1.2$	weight for violating a time window by one time unit
$\sigma_1 = 33$	Score if new global best solution was found
$\sigma_2 = 18$	Score if new and unvisited solution was found with better objective function value than current solution
$\sigma_3 = 9$	Score if new and unvisited solution, not better than current objective value, but still accepted
$\tau = 50$	Number of iterations per segment, number of iterations before probability update of operators
$\omega = 25,000$	Number of total iterations
$\Omega = 0.5$	Parameter for acceptance of initial solution; used to calculate start temperature
$c = 0.99977$	Cooling rate
$c^{\text{na}} = 5 \cdot \max_{i,j}(c_{i,j}^{\text{travel}})$	Costs of not assigning a customer
$c^{\text{pred}} = 8.7$	Costs of violating a precedence relation
$c^{\text{skill}} = 3.4$	Costs of violating a skill level by one time unit
$c^{\text{tw}} = 1.8$	Costs of violating a time window by one time unit
$\max(n^{\text{bt}}) = 5000$	Maximum number of backtracking steps.
$r = 0.2$	Reaction parameter (roulette parameter)
t^{Percent}	Auxiliary parameter to determine the end temperature
$q_1^{\text{lb}} = 0.1$	Auxiliary value to determine lower bound on number of nodes that are removed from current solution
$q_2^{\text{lb}} = 30$	Auxiliary value to determine lower bound on number of nodes that are removed from current solution
q^{lb}	Lower bound on number of nodes that are removed from current solution. $q^{\text{lb}} = \min\left\{\lvert\mathscr{I}\rvert \cdot q_1^{\text{lb}},\ q_2^{\text{lb}}\right\}$
$q_1^{\text{ub}} = 0.4$	Auxiliary value to determine upper bound on number of nodes that are removed from current solution

ALNS parameters in alphabetical order (continued)

$q_2^{\text{ub}} = 60$	Auxiliary value to determine upper bound on number of nodes that are removed from current solution
q^{ub}	Upper bound on number of nodes that are removed from current solution. $q^{\text{ub}} = \min\left\{\lvert\mathscr{I}\rvert \cdot q_1^{\text{ub}},\ q_2^{\text{ub}}\right\}$
T^{start}	Start temperature $T^{\text{start}} = -\frac{\Delta}{\ln \Omega} \cdot f(s_0)$
T^{end}	End temperature $T^{\text{end}} = T^{\text{start}} \cdot t^{\text{Percent}}$
$p^{(op)} = 9$	Control parameter for the randomness of the removal operators
10	Number of features for penalty update
25	Iterations between penalty updates
1.1	Penalty initial value
0.69	Penalty increase if feature is violated
0.08	Penalty reduction if feature is not violated
3	Maximum time window violation expressed in time units
100	Number of solutions that are considered when calculating the request graph. Needed for request graph (historic) destroy operator.

7

Social Media Graphs

We're all in this together if we're in it at all.

Johnny Cash
1932-2003

Attribution

This Chapter uses material from the following publication:

- Klaus Arthur Schmid, Christian Frey, Fengchao Peng, Michael Weiler, Andreas Züfle, Lei Chen, and Matthias Renz. Trendtracker: Modelling the motion of trends in space and time. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1145–1152. IEEE Computer Society, 2016. doi: 10.1109/ICDMW.2016.0165. URL `https://doi.org/10.1109/ICDMW.2016.0165`. [183]

See § 1.4 for an overview of incorporated publications and the author's attribution.

Highlights

- Modeling trends of a microblogging system as a 4-dimensional trend-flow tensor;
- Identification of archetypes of dissemination of trend flows;
- Evaluation shows various trend-archetypes such as political trends, disaster trends and celebrity trends;

7.1 Machine Learning for Analyzing Social Media Data

In the late 1990s, the first social network applications (e.g., Classmates, SixDegrees) were published. Since then, social networks such as Facebook, Twitter, and Instagram, among others, have attracted billions of users worldwide. Nowadays, many integrated interactions with other people via those platforms in their daily lives to talk about topics ranging from simple entertainment, to politics to recent advances in various research fields. Social media not only brought up a new way of communication but also drastically changed the spread of information and opened up new business branches. Users can share ideas and opinions, post updates and feedback, engage in activities and events, share their interests, offer products, communicate job offers, and so forth. The most natural way of modeling the interactions in a social network is provided by *Graph Theory*. The vertices in a graph correspond to persons, and edges represent any interaction two persons can have with each other, e.g., following each other, giving feedback in the form of likes, sharing posts in a thread, and so forth. Hence, graphs provide a very powerful tool to model and finally extract valuable knowledge about user behaviors, interpret interactions among single persons or communities, or integrate users' interests in a recommendation system. Due to the nature of ever-growing social networks, we are faced with highly complex problems in understanding the dynamics arising in those highly active platforms.

This chapter presents a model to analyze the dynamics of trends in a social network using machine learning. The first step is to extract the topics contained in a text database. Due to the users' location affiliation, the texts sent to the social media platform can be geo-tagged. Hence, we are given information about trends and statistically significant topics arising in a specific location. Furthermore, the messages are sent at a certain time. Therefore, we can follow the messages in two dimensions, in the spatial as well as the temporal domain. The problem discussed in the following is identifying archetypes referring to disseminating trends over space and time. Given the natural mapping of graph representations and their matrices (cf. § 2.2.2), we can model the spatial-temporal trend flow in a 4-dimensional tensor, where two dimensions refer to the spatial locations, one to the temporal domain, and one to the various trends. In the language of graph theory, the trend flow, which shows the flow of one piece of information to another, represents two connected vertices. Additionally, we are given the amount of information flow as attributes on the edges. Moreover, the connection also contains information about the temporal occurrence and the topic which arise in both locations.

7.2 Tracking Trends in Social Media

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-textual data. Such data includes microblogging

platforms such as Twitter, social networks such as Facebook, and data from news stations. Such geo-textual data allows to immediately detect and react to new and emerging trends. A trend is a set of keywords associated with a time interval where the frequency of these keywords is increased significantly.

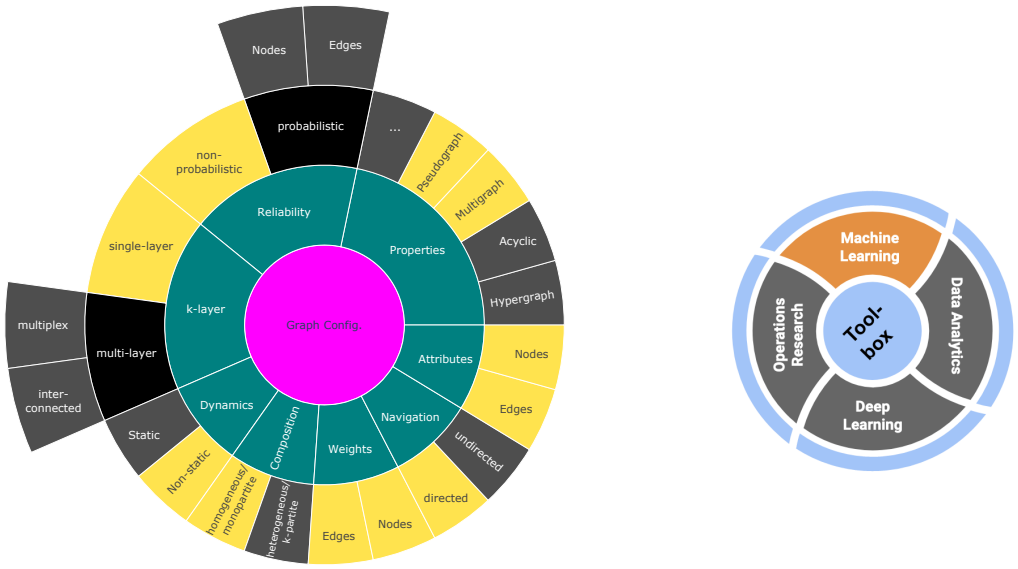
In this chapter, the thesis investigates the dissemination of trends over space and time. For this purpose, a four-step framework is proposed. In the first step, the model employs existing solutions to mine a large number of trends. Second, for each trend a spatio-temporal dissemination model is created, which describes the motion of this trend over space and time. To model this dissemination, a (flow-source, flow-destination, time, trend)-tensor is employed. In the third step, the model clusters these trend-tensors, to identify groups of archetype trends. For each archetype, all tensors of the same archetype are aggregated and a tensor factorization approach is employed to describe this archetype by its latent features. As the fourth step, an algorithm is proposed which can classify the trend-archetype of a new trend, in order to predict the future dissemination of this trend.

The experiments reveal that the space of trends does exhibit clusters, each corresponding to a trend-archetype such as political trends, disaster trends and celebrity trends. Furthermore, it is shown that by identifying the trend-archetype of a trend, we can effectively predict the future of this trend.

Graph Configuration.
The graph's configuration for the analysis of trends flows in social media is shown in Figure 7.1a. The setting is a single-layer, non-probabilistic, non-static, homogeneous graph. The occurrence of topic mentions is expressed as weights on the nodes, and weight values on the edges express the trend dissemination between two locations. Moreover, attributive information on the edges indicate the relevant topic, whereas attributive information on the nodes indicate the spatial dimensions of the respective locations. Because of permitting multiple relationships between pair of nodes, we are in a setting of a multigraph, and by allowing self-loops we have a pseudograph. Even though, the methodology is primarily discussed using a 4-dimensional tensor, the mapping to a graph is relatively trivial (cf. Matrix Representations § 2.2.2). The technical tools are affiliated to the scope of *Machine Learning*.

7.2.1 Motivation

Social media such as Twitter or other microblogging platforms are a popular source for live textual data, often associated with geographic information. Such data may describe an event, an experience or a point of interest that is relevant to a user. More generally speaking, such microblogs describe events, objects and persons that are on the mind of a user. The prediction of trends has a plethora of economic applications in targeted marketing and investment banking, by knowing what people will have on their mind tomorrow. In this chapter, the aim is not to



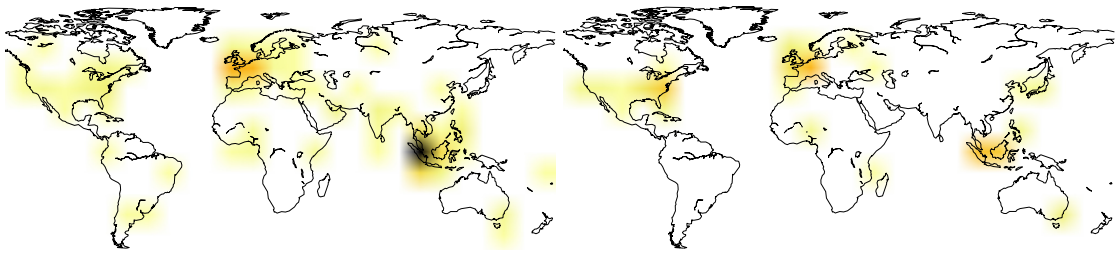
(a) Graph Configurator *(b) Toolbox*

Figure 7.1: *Graph Configuration and methodology's classification used to analyze trend dissemination in social networks by TrendTracker*

predict new trends. However, the proposed model predicts the flow of existing trends over the globe. For instance, trends related to *fashion* might often arise in France, then move over to the rest of Europe within a few days, then start to affect North America within weeks, and then flow to Australia within weeks and months. In contrast, technological trends might often be initiated in Japan and South Korea, then flow to North America, and only then flow to Europe.

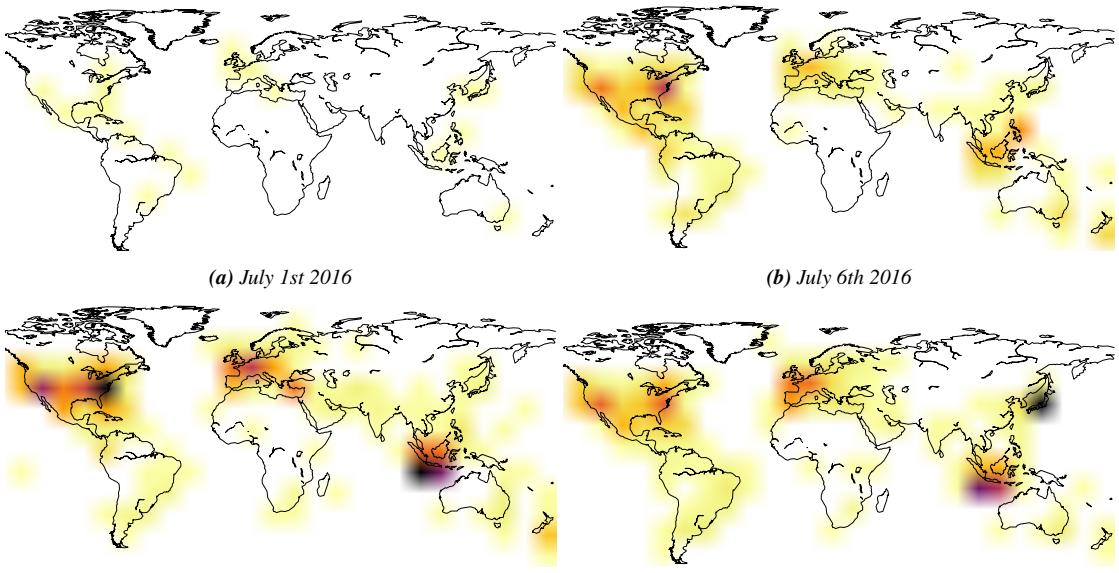
As an example of such a trend, Figure 7.2 shows the location of tweets issued in July of 2014 corresponding to the lost Malaysian Airlines flight "MH17". The trend shows initial strong bursts in Malaysia as well as in the Netherlands, from where the missing flight originated, as seen in Figure 7.2a. From there, the trend quickly spread all across the world – two days later, the rest of Europe as well as North America are just as involved in the trend. This can be seen in Figure 7.2b.

Another trend development can be seen in Figure 7.3, where the location of tweets containing the string "Pokémon" is shown for several days. Beginning with the first of July, 2016, Figure 7.3a exhibits a globally low interest in this topic, indicating no trend at that time. As the free-to-play game "PokémonGo!" was released for cell phones in the United States, Figure 7.3b shows a highly significant burst of tweets on this topic on July the 6th, originating in the US alone. One week later, on July 13th, the trend has moved to Europe as the game was released in several countries there. This can be seen in Figure 7.3c. Asia follows, mainly with the Japan release on July 22nd, with a high activity regarding the topic as shown in Figure 7.3d.



(a) July 17th 2014 *(b) July 19th 2014*

Figure 7.2: *Distribution of trend "MH17"*



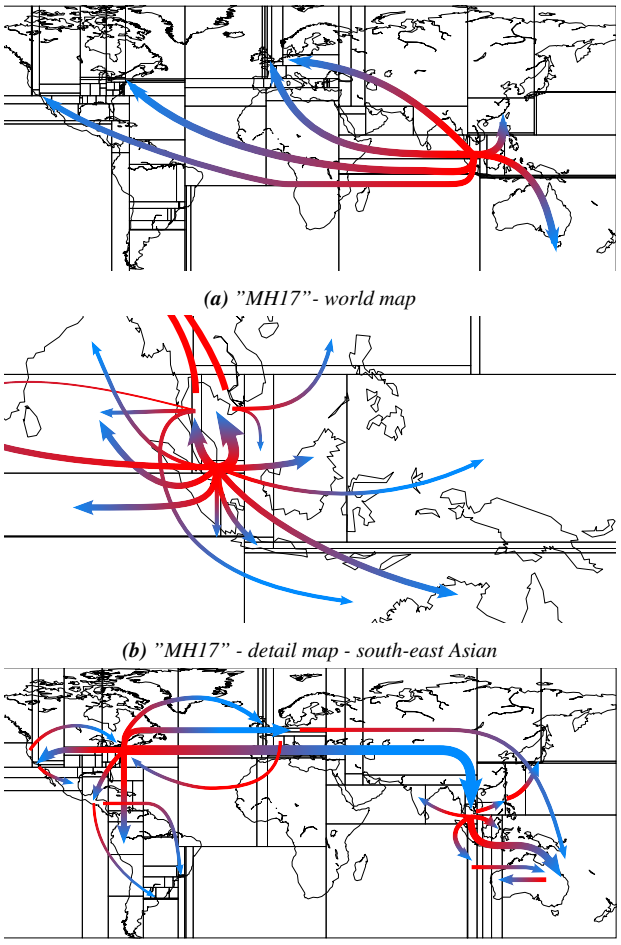
(a) July 1st 2016 *(b) July 6th 2016*

(c) July 13th 2016 *(d) July 22nd 2016*

Figure 7.3: *Distribution of trend "PokémonGo!"*

Intuitively, different types of trends are expected to show different distributions. While a few trends spread to a global scale within hours due to dissemination through news networks, other trends may be more local, spread slower, might be originating from specific regions, or might disseminate to specific regions only.

In the following, the proposed method models and mines such dissemination of trends over space and time. That is, the model observes the flow of trends, specified by source and target regions, over time. Figure 7.4 exemplifies such flows for the two examples given before, namely "MH17" and "PokémonGo!". The arrows on the map indicate a flow in activity from source (red) to target (blue). For the sake of readability, the representation has been kept coarse and omits certain regional interdependencies. Geographical regions are referenced by their position in the index (drawn in black outlines), and thickness of arrows indicates strength of the dependence. Figures 7.4a and 7.4b exhibit trend dissemination of the trend "MH17" in a full world view and one of the south-east Asian region alone, respectively. As can be seen very clearly,



(a) "MH17"- world map

(b) "MH17" - detail map - south-east Asian

(c) "PokémonGo!" - world map

Figure 7.4: *Spatio-Temporal Trend Dissemination*

the trend originates from Malaysia and spreads over the world from there, partially using other regions as intermediate hops. In contrast, Figure 7.4c uses the same representation for the trend "PokémonGo!" on a world-wide scale and while there is a general main direction from the US east coast, several rules in the opposite direction indicate a more diverse dissemination pattern. Curiously, once again, south-east Asia is a strong hub for this trend, resulting from a local burst on this topic from Indonesia.

But rather than looking at a few, hand-selected, trends as shown in these figures, existing trend mining solutions are used to automatically extract the disseminations of a large number of past trends. Each trend yields a spatio-temporal trend-tensor, containing for each discrete time interval, and each spatial region the number of corresponding tweets. As a first contribution, the thesis postulates and verifies the hypothesis that trends follow different archetypes, which differ strongly in terms of their dissemination patterns. Using a clustering approach, the model identifies these archetypes trends. For new trends, this result can be used to quickly classify a

new trend as an archetype trend, to more effectively predict its future dissemination, allowing to predict where a trend will move to in the near future.

To model the dissemination of trends in space and time, the chapter is organized as follows. The next section, Section 7.2.4 gives an overview over the state-of-the-art of modeling trends in space and time. Section 7.2.2, formally defines a trend, and introduces the notion and data structures to define the spatio-temporal motion of a trend. Section 7.2.3.4 presents the technical concept for modeling the dissemination of a trend. This concept is experimentally evaluated in Section 7.2.5, and the chapter concludes in Section 7.3.

7.2.2 Preliminaries

This section defines terms and notations used throughout this chapter, and formally defines the problems tackled in the following. In this thesis, spatio-temporal text data is considered, that is text data annotated with a geo-location and a timestamp, such as obtained from Twitter.

Definition 28 (Spatio-Temporal Text Database)**.** *A* spatio-temporal text database $\mathscr{D}$ *is a collection of triples* (s,t,c)*, where* s *is a point in space,* t *is a point in time, and* c *is a textual content.*

A concept that is adapted from the literature is the concept of a *trend* as introduced in [185].

Definition 29 (Trend)**.** *A trend* $\tau_{K,t}$ *is a set of keywords* K *that appear significantly more often starting at a time* t*.*

A more formal definition, which introduces the requirements of a set of terms to be considered as significant, is given in Section 7.2.3.1. The set of spatio-temporal text objects which support trend $\tau_{K,T}$, is denoted as

$$\mathscr{D}_{\tau_{K,T}} = \{(s,t,c) \in \mathscr{D} | c \in K \wedge t \in T\}.$$

Definition 30 (Spatio-Temporal Occurrence)**.** *Let* $\tau_{K,T}$ *be a trend. Let* $\mathscr{S} = \{S_1, ..., S_{|S|}\}$ *be a partitioning of space into spatial regions, and let* $\mathscr{T}$ *be a partitioning of time into equi-sized time intervals denoted as epochs. Further, let* $T := t \cap \mathscr{T} = \{T_1, ..., T_{|T|}\}$ *be the set of epochs overlapping the trending time* T*. Then*

$$Occ_{\tau_{K,T},S} = |\{(s,t,c) \in \mathscr{D} | s \in S \wedge t \in T \wedge c \in K\}|.$$

is the number of occurrences of trend $\tau_{K,T}$ *at region* S*.*

The aim of this thesis is to find the dissemination of trends, that is, pairs of spatial locations (S_1, S_2) such that any trend that appears in region S_1 is significantly more likely to appear in S_2 in the next epoch.

To describe the motion of a trend (K,t) in space and time, each trend is described by a time-space matrix, describing for each spatial region and each epoch $t \in T$ the number of tweets of the trend.

Definition 31 (Trend Count Matrix)**.** *The trend count matrix $D(\tau_{K,T}) \subseteq R^{|S|} \times R^{|T|}$ contains all occurrences of trend $\tau_{K,T}$ over space and time, and is defined as follows:*

$$D(\tau_{K,T})_{i,j} = Occ(\tau_{K,T_i}, S_j)$$

In this model, the main task is to analyze and mine multiple trend count matrices as defined in Definition 31, in order to identify groups of similar trends, groups of similar spatial regions, and to find common spatio-temporal dissemination of trends. These problems are formally defined as follows.

Definition 32 (Trend Clusters)**.** *Let $\mathscr{D}$ be a spatio-temporal text database, let $\mathscr{D}_\tau$ be a set of trends mined from $\mathscr{D}$, and let $D(\tau \in \mathscr{D}\tau)$ denote the trend count matrix of each trend. A trend cluster $C \subseteq \mathscr{D}_\tau$ is a set of trends that exhibit mutually similar trend count matrices.*

Given a set of trends, the main challenge is to find association rules of the form *"Any trend observed in region A today, is likely to appear in region B tomorrow"*. This kind of spatio-temporal trend dissemination is defined as follows.

Definition 33 (Spatio-Temporal Trend Dissemination Rule)**.** *Let $\mathscr{D}_\tau$ be a set of trends and their corresponding trend count matrices $D(\tau \in \mathscr{D}\tau)$. For two spatial regions S_s and S_t, a spatio-temporal trend dissemination rule $S_s \rightarrow S_t$ implies that a large trend count at source region S_s at any time t indicates a large trend count at target region S_t at time $t+1$, formally:*

$$(S_s \rightarrow S_t) \leftrightarrow \forall i, \forall \tau \in \mathscr{D}_\tau : D(\tau)_{i,s} \rightarrow D(\tau)_{i+1,t},$$

where $D(\tau)_{i,s} \rightarrow D(\tau)_{i+1,t}$ denotes that a large value in $D(\tau)_{i,s}$ implies a large value in $D(\tau)_{i+1,t}$

Finally, Definition 33 allows to define the problem of spatio-temporal trend dissemination rule mining.

Definition 34. *Let $\mathscr{D}_\tau$ be a set of trends and their corresponding trend count matrices $D(\tau \in \mathscr{D}\tau)$. The problem of spatio-temporal trend dissemination rule mining is to find all pairs of spatial regions (S_s, S_t) such that $(S_s \rightarrow S_t)$ holds.*

7.2.3 Spatio-Temporal Trend Dissemination Rule Mining

This section describes the novel approach at mining spatio-temporal trend dissemination rules. As a first step, we need to acquire past trends, to mine dissemination rules from. For this purpose,

existing textual trend mining solutions proposed in the recent past are applied, which are briefly sketched in Section 7.2.3.1 for self-containment. Next, as a second step used for preprocessing, a space composition scheme is employed in Section 7.2.3.2 to ensure having a similar number of tweets in each spatial region using a k-d tree. As a third step, the flow of trends over space and time is modeled in Section 7.2.3.3. Therefore, a *trend count matrix*, as defined in Definition 31, is transformed into a *trend flow tensor*, which describes the flow from any source region to any target region at any point in time for any trend. Consequently, constructing a *trend flow tensor* for each trend that was mined in the first step, yields a four-mode Space $\times$ Space $\times$ Time $\times$ Trends tensor, which will be fed to the fourth step, the mining step. In the mining step proposed in Section 7.2.3.4, a tensor factorization approach is employed to discover latent features of trends, latent features of trend-source-regions and latent features of trend-target-regions. These latent features allow to cluster trends into sets of trends which disseminate similarly over space and time. Then, for each cluster of similar trends, trend flows are obtained from the *reconstructed trend flow tensor*.

7.2.3.1 Traditional Trend Mining

The model uses SigniTrend [185] to establish a trend baseline. SigniTrend uses Count-min data structures [35] for approximate counting and tracks the average and standard deviation of term and term pair frequencies. In order to estimate the average $EWMA$ and the variance $EWMVar$ for a frequency x on a data stream, they can rely on earlier work by Welford [218] and West [221] on incremental mean and variance. The update equations given by Finch [58] for the exponentially weighted variants allow these values to be efficiently maintained on a data stream:

$$\begin{aligned} \Delta &\leftarrow x - EWMA \\ EWMA &\leftarrow EWMA + \alpha \cdot \Delta \\ EWMVar &\leftarrow (1-\alpha) \cdot (EWMVar + \alpha \cdot \Delta^2) \end{aligned}$$

The learning rate α can be set using the half-life time $t_{1/2}$; a parameter a domain expert will be able to choose easily based on his experience and needs:

$$\alpha_{\textit{half-life}} = 1 - \exp\left(\log\left(\tfrac{1}{2}\right)/t_{1/2}\right)$$

To capture interesting relationships among trends (such as `"Facebook"` bought `"WhatsApp"` or Edward `"Snowden"` traveled to `"Moscow"`) SigniTrend also tracks word pairs. A single term is thereby modeled as a co-occurrence with itself. Given a word pair (w,l) where w and l are single word tokens, SigniTrend uses a classic model from statistics to measure the significance: Let $f_t(w,l)$ be the relative frequency of this pair of tokens within the documents $D_t = \{d_1, \ldots, d_n\}$ at

time t, i.e.:

$$f_t(w,l) := \frac{|\{w \in d \wedge l \in d \mid d \in D_t\}|}{|D_t|} \tag{1}$$

then they use the series of previous values $f_1, \ldots, f_{t-1}$ to compute an estimated value and a standard deviation. To facilitate aging of the data and to avoid having to store all previous values, they employ the exponentially weighted moving average ($EWMA[f(w,l)]$) and moving standard deviation ($EWMVar[f(w,l)]$). With these estimates, the z-score of the frequency is computed as follow:

$$z_t(w,l) := \frac{f_t(w,l) - \max\{EWMA[f(w,l)], \beta\}}{\sqrt{EWMVar[f(w,l)]} + \beta} \tag{2}$$

The term β is motivated by the assumption that there might have been $\beta \cdot |D|$ documents that contained the term, but which have not been observed due to incomplete data. With this Laplace-style smoothing we prevent instability for rare observations of pairs (w,l). For Twitter, the suggested value for this term is $\beta = 10/|D|$: intuitively we consider 10 occurrences to be a by chance observation. This also adjusts for the fact that we do not have access to the full Twitter data.

Terms and pairs with corresponding z-scores (see Equation 2) larger than a given threshold τ are considered as trends. For the experiments shown in this thesis, $\tau = 3$ is used.

7.2.3.2 Space Decomposition Scheme

To fit a flow model between spatial regions, we need to minimize the bias that results from having a non-uniform distribution of tweets on earth. The model remedies this problem by partitioning the geo-space in a way that minimizes the difference of tweets between spatial regions. For this purpose, the geo-locations of all tweets in the input database are inserted into a k-d tree, having a maximum node capacity of 1000. Thus, every leaf node of this k-d tree is guaranteed to have between 500 and 1000 two-dimensional points. Each of this leaf node is then used as a spatial region in the remainder of this thesis. The decomposition that is obtained this way is exemplarily depicted in Figure 7.5. Note that this tree is constructed upon a typical, yet static, set of tweets.

7.2.3.3 Trend Flow Modeling

In this section, the novel approach of obtaining a trend flow from raw trend s is described. Thus, for a given trend, all N occurrences of this trend at some time t are considered and all M occurrences at the next time $t+1$. All the regions having the trend at time t can be considered as sources of the trend, and all regions having the trend at time $t+1$ can be considered as targets of the trend. Yet, we do not know any more specifically, which source region has affected which target region and to what degree, since we do not know through which channels and medias the

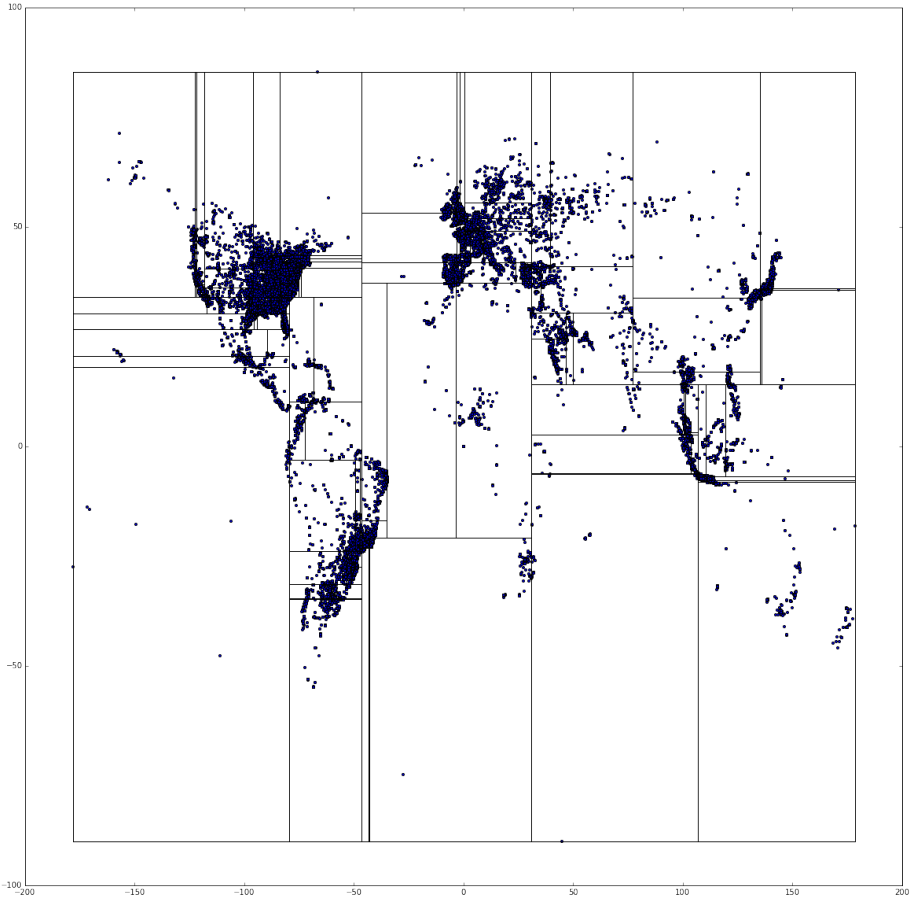


Figure 7.5: *k-d tree based space decomposition*

trend was disseminated. Thus, due to lack of better knowledge, the assumption holds that all sources affect all target uniformly. This flow model is formalized as follows:

Definition 35 (Spatio-Temporal Trend Flow Model)**.** *Let $\tau_{K,T}$ be a trend having a set of keywords K and having a time interval $T = \{T_1, ..., T_{|T|}\}$ which covers $|T|$ epochs. Let $\mathscr{S} = \{S_1, ..., S_{|\mathscr{S}|}\}$ be a space composition into $|\mathscr{S}|$ spatial regions. Furthermore, let $D(\tau_{K,T})_{i,j} = Occ(\tau_{K,T_i}, S_j)$ be the trend matrix of $\tau_{K,T}$. We define the* trend flow model *$F(\tau_{K,T})$ of trend $\tau_{K,T}$ as a $\mathscr{S} \times \mathscr{S} \times \{T_1, ..., T_{|T-1|}\}$ tensor, such that*

$$F(\tau_{K,T})_{i,j,k} = \frac{Occ(\tau_{K,T_k}, S_i) \cdot Occ(\tau_{K,T_{k+1}}, S_j)}{\sum_{S_n \in \mathscr{S}} Occ(\tau_{K,T_{k+1}}, S_n)}$$

Intuitively, an entry $F(\tau_{K,T})_{i,j,k}$ of tensor $F(\tau_{K,T})$ corresponds to the absolute flow of occurrences from region S_i to region S_j from time T_k to time T_{k+1}.

Example 3. *To illustrate the construction of tensor $F(\tau_{K,T})$, consider an example depicted in Figure 7.6. Here, the occurrences matrix of a tensor of a trend is shown for four spatial regions.*

$T_i \to T_{i+1}$

$$\begin{pmatrix}2\\0\\0\\1\end{pmatrix} \xrightarrow[S_s \to S_t]{flow} \begin{pmatrix}3\\1\\0\\5\end{pmatrix} \xRightarrow[F(\tau_{K,T})]{tensor} \begin{pmatrix}\frac{6}{9} & \frac{2}{9} & 0 & \frac{10}{9}\\ 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ \frac{3}{9} & \frac{1}{9} & 0 & \frac{5}{9}\end{pmatrix}$$

$T_{i+1} \to T_{i+2}$

$$\begin{pmatrix}3\\1\\0\\5\end{pmatrix} \xrightarrow[S_s \to S_t]{flow} \begin{pmatrix}2\\1\\3\\4\end{pmatrix} \xRightarrow[F(\tau_{K,T})]{tensor} \begin{pmatrix}\frac{6}{10} & \frac{3}{10} & \frac{9}{10} & \frac{12}{10}\\ \frac{2}{10} & \frac{1}{10} & \frac{3}{10} & \frac{4}{10}\\ 0 & 0 & 0 & 0\\ 1 & \frac{5}{10} & \frac{15}{10} & 2\end{pmatrix}$$

Figure 7.6: *Trend Flow Modeling*

At the first point of time t_i, the trend appears twice in the first region and once in the fourth region, yielding the vector $(2,0,0,1)^T$. The second t_{i+1} and third point of time t_{i+2}, the distribution of occurrences is $(3,1,0,5)^T$ and $(2,1,3,4)^T$, respectively, yielding the trend matrix shown in Figure 7.6. Transitioning from the first epoch t_i to the second epoch t_{i+1}, the occurrences change from $(2,0,0,1)^T$ to $(3,1,0,5)^T$. The first spatial location S_1, having an initial value of two tweets, is thus a source of the trend. Since we cannot observe the latent means of dissemination of a trend (through the internet, via TV, radio, word-of-mouth, etc.), one can estimate that region S_1 disseminates its trend to all other regions having this trend. Since a fraction $\frac{3}{9}$ of all tweets at time t_{i+1} are observed in region S_1, we estimate a trend-from of $\frac{2\cdot3}{9}$ from region S_1 to itself. In contrast, only one trending tweet is observed at location S_2 at time t_{i+1}, of which we contribute a flow of $\frac{2\cdot1}{9}$ to S_2. Similarly, a flow of $\frac{1\cdot5}{9}$ is contributed from S_4 to S_4.

It is notable that each time-slice of tensor $F(\tau_{K,T})$ is a rank-1 matrix, as all lines are multiples of each other. This redundancy is desirable, as it evenly distributes the flow from all source regions to all target regions, and this redundancy will be removed in a later tensor factorization step. For each trend $\tau_{K,T}$ a three-mode tensor as described in Definition 35 is obtained. Concatenating these tensors for each trend $\tau \in \mathscr{D}_\tau$ yields a four-mode tensor $\mathscr{F}(\mathscr{D})$ which is passed into the trend flow mining step described in the next section.

As insinuated in § 7.2, the trend flow model can also be interpreted as a graph:

Definition 36. *The pseudograph $\mathscr{G} = (\mathscr{S}, \mathscr{E}, \phi, \psi)$ is a directed multigraph with self-loops, where $\mathscr{S}$ denotes the set of vertices for the locations and $\mathscr{E}$ denotes the labeled edge set reflecting the trend flow between locations. The function psi : $\tau_{K,T} \times \mathscr{V} \to \mathbb{R}$ yields the occurrence of trend $\tau_{K,T}$ for a specific location as node weight. The edges in the graph indicate the flow of trend $\tau_{K,T}$ from a region $S_i \in \mathscr{S}$ to region $S_j \in \mathscr{S}$ from time T_k to time T_{k+1}. We define a function*

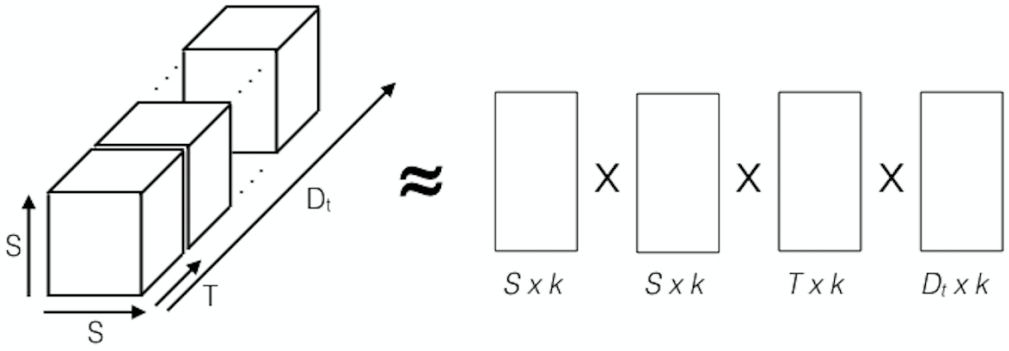


Figure 7.7: *Trend Flow Modeling - Tensor Decomposition*

$\phi : \tau_{K,T} \times \mathscr{E} \to \mathbb{R}$ *yielding the information of* $F(\tau_{K,T})_{i,j,k}$ *as edge weight.*

Recall, a graph is a multigraph if multiple edges are allowed between vertices for modeling several trends between locations. It is also called a pseudograph if self-loops are allowed. As discussed in Figure 7.6, the source location of a trend can also be the destination of it.

7.2.3.4 Trend Flow Mining

The thesis proposes to decompose tensor $\mathscr{F}(\mathscr{D}) \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ using a CANDECOMP/PARAFAC tensor decomposition [27], [81] using k latent features, where k is a parameter of the algorithm. A *CP* factorization decomposes a tensor into a sum of component rank-one-tensors, i.e.:

$$\mathscr{F}(\mathscr{D}) \approx \sum_{r=1}^{k} u_r^1 \circ \cdots \circ u_r^N$$

where $u^n \in \mathbb{R}^{I_n}$ for $n = 1, \ldots, N$. Hence, as illustrated in Figure 7.7, this factorization decomposes the four-mode $\mathscr{S} \times \mathscr{S} \times T \times \mathscr{D}_\tau$ tensor into four sets of vectors:

- a set of k vectors of latent features of length $|\mathscr{S}|$ describing each source spatial region,
- a set of k vectors of latent features of length $|\mathscr{S}|$ describing each target spatial region,
- a set of k vectors of latent features of length $|T|$ describing each time epoch, and
- a set of k vectors of latent features of length $|\mathscr{D}_\tau|$ describing each trend.

These k-dimensional feature vectors can be used to identify mutually similar source spatial regions, mutually similar target spatial regions, mutually similar points in time, and mutually similar trends.

7.2.3.5 Trend Archetype Clustering

In the first mining step, clusters of mutually similar trends are identified, i.e., trends which have a similar feature vector after the factorization and, thus, since the tensor $\mathscr{F}(\mathscr{D})$ describes the flow of trends over time, exhibit a similar dissemination over space and time. Each of the resulting clusters is called a trend archetype. This approach allows to classify future trends among all archetypes, and allows to predict the future dissemination of a new trend by using the dissemination model of their archetype.

Definition 37. *Let $\mathscr{D}_\tau$ be a set of trends, and for each trend $\tau \in \mathscr{D}_\tau$ let feat(τ) be a set of features describing τ. Further, let $\mathscr{C}(\mathscr{D}_\tau) = \{C_1, \ldots, C_n\}$ be a clustering of all trends in $\mathscr{D}_\tau$ into n clusters. Then we denote each cluster $C \in \mathscr{C}$ as an archetype, and all trends $\tau \in C$ are said to belong to the same archetype.*

7.2.3.6 Trend Archetype Flow Modeling

After the trend clustering step of Section 7.2.3.5, sets of trends are identified which belong to the same dissemination archetype. Therefore, we return to the full tensor $\mathscr{F}(\mathscr{D})$, and for each archetype $C \in \mathscr{C}$, we select only the trends $\tau \in C$, thus, yielding a $\mathscr{S} \times \mathscr{S} \times T \times C$ tensor $\mathscr{F}(\mathscr{D}, C)$ for each archetype C. Using $\mathscr{F}(\mathscr{D}, C)$, a projection on two modes $\mathscr{S} \times \mathscr{S}$ is performed by averaging over all trends $\tau \in C$ and all epochs $T_i \in T$ to obtain the flow model of archetype C.

7.2.4 Related Work and Discussion

The problem of event detection in social media streams has attracted much attention in recent years. Ritter et al. [170] developed an event extraction system based on Twitter streams. Using the entity recognition and sequence classification, they extracted a 4-tuple representation of each event, showing the entities, mentions, calendar, and type of each event. Schubert et al. [185] proposed a statistical metric based on the term frequency, and reported an event when there was a large deviation in the metric of a particular term. They applied hierarchical clustering to merge terms that burst together into large-scale topics. In addition to textual information, Kalyanam et al. [104] also considered the communities of users who are interested in certain topics. They applied non-negative matrix factorization (NMF) to incorporate both textual and social information in studying the topic detection and evolution. Lin et al. [139] applied a Gibbs Random Field model regularized by a topic model to track the popular events in social media. For each evolving event, they reported a stream of text information and a stream of network structures indicating the event diffusion. Weng et al. [220] applied Wavelet Transform to build signals for each word. Then they built a graph based on the cross-correlation of signals and clustered words into events using a modularity-based graph partitioning technique. Sayyadi [181] et al. applied community detection technique to detect events in social streams. They built a graph of words

based on their co-occurrence. Then they removed the vertices with high betweenness centrality score and regarded the communities that remained as the keywords for events.

However, none of these works exploited the spatio-temporal characteristics of an event. Unankard et al. [205] extracted user locations and event locations from geo-tagged posts. They defined a location correlation score between user and event locations and used it to identify the hotspot events. Zhou et al. [233] extended the Latent Dirichlet Allocation (LDA) to incorporate the location information of social messages, and proposed a novel location-time constrained topic model. Then they detected events by conducting similarity joins in streams of social messages. Sakaki et al. [177] conducted semantic analysis in user posts to detect natural disasters. They used exponential distribution to study the temporal characteristics of disasters. They used kalman filter and particle filter to predict the spatial trajectories of disasters. From the perspective of query processing, Lappas et al. [126] defined two types of spatio-temporal burstiness patterns, aiming at finding terms which had unusually high frequencies in a spatial region within a particular time interval. Sankaranarayanan et al. [178] developed a news system based on Twitter streams. They used Naïve Bayes Classifier to distinguish valuable news from junk posts and used an algorithm called leader-follower clustering to cluster news into topics. Appice et al. [4] proposed a technique where trend clusters are used to summarize sensor readings. However, such clusters consist of sensor entities themselves as opposed to trends.

7.2.5 Evaluation

7.2.5.1 Parameters and dataset

The proposed workflow is evaluated on a dataset mined from Twitter using their public API, feeding from a global 1%-sample over the years 2014 through 2016 (until August of 2016). Out of the tweets returned from the API, those without a geolocation specified are removed. Tweets were aggregated over one-day periods by their UTC-timestamp. The number of tweets per day ranged from around 50,000 to 150,000.

For each trend from the SigniTrend framework, tweets were extracted from one day before and five days after the respective associated date to cover the entire trend dissemination pattern. Unless otherwise specified, each day was subdivided into epochs of six hours to allow for timeshift in different hemispheres. For the majority of the experiments, the top-100 trends of the year 2014 were used.

7.2.5.2 Evaluation of trend archetypes

Table 7.1 depicts some exemplary resulting trend archetypes from data covering the year 2014. Keywords for the top-100 trends were extracted using SigniTrend and used to filter geo-tagged tweets occurring within a 5 day timeframe around the trend date. Underscores "_" between words denote a boolean conjunction, requiring all connected words to occur in any possible order within

Table 7.1: *Trend Archetypes of 2014*

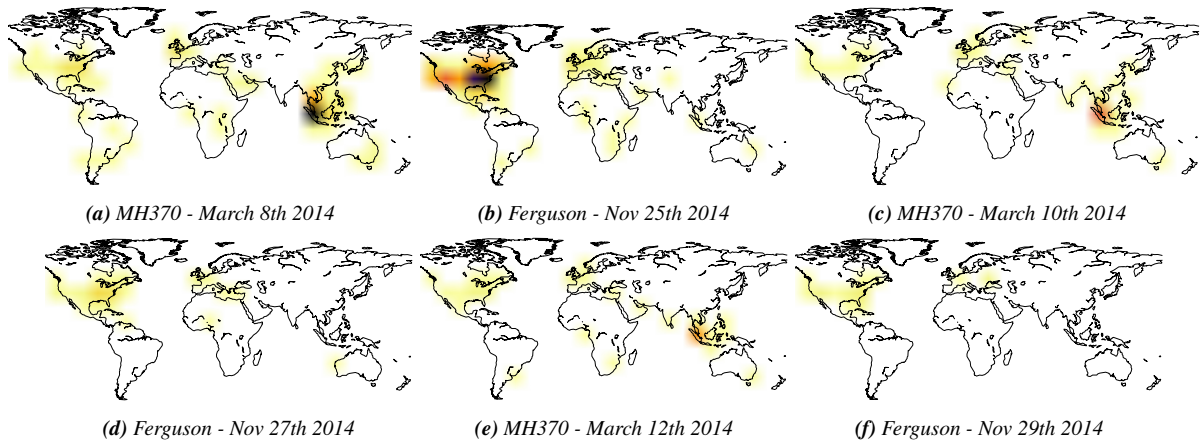
#	Size	Example 1 Keywords	Example 2 Keywords
1	8	mh17 malaysia_crash	ferguson michael_brown riot
2	3	ellen degeneres selfie	robin williams suicide
3	5	whatsapp facebook takeover	supreme_court obergefell hodges
4	10	germany fifa14 brazil	germany fifa14 argentina
5	4	brazil world_cup	ebola
6	12	eu_sanction eu_russia	putin peskov conference
7	1	chile iquique earthquake	-
8	10	flappy_bird removed_appstore	how_I_met_your_mother_finale
9	18	mh370 malaysia_missing	qz8501 air_asia missing
10	14	scotland independence_poll	india bharatiya janata election
11	14	sydney siege hostage	ottawa gunman parliament
12	1	merry chistmas	-

one tweet. Spaces between keywords or conjuctions of keywords denote a boolean disjunction. Keywords listed are not exhaustive.

Each line of the table corresponds to a resulting archetype of trends with similar dissemination, resulting from a clustering of the latent feature vector feat(τ). While column "Size" refers to the true cardinality of each cluster, (up to) two examples are given to illustrate the nature of each archetype. Each example lists some keywords for one trend grouped into this archetype.

Some rather interesting results emerge by comparing the keywords to their respective historical events. While archetype #9 contains two trends referring to airplanes going missing without a trace (MH370 in March and QZ8501 in December), another lost airplane is grouped together with riots in the aftermath of a police shooting in the US in archetype #1. Looking at the respective tweet heatmaps in Figure 7.8, a similarity in pattern emerges: a first main event occurs ("plane crashes in Ukraine" vs. "riots after jury decision not to indict shooter") causing an initial burst mainly in the affected areas (Figures 7.8a for MH370 and 7.8b for the shooting). After the initial burst, new information sheds different light on the events, making them stand out and causing a more steady flow of messages internationally ("plane grounded by missile" vs. "several people killed as riots spread"). This more steady output can be seen over Figures 7.8c and 7.8e for MH370 and Figures 7.8d and 7.8f for the shooting. Bear in mind that the grouping occurred solely based on the numerical features of the respective trends' spatial dissemination, regardless of their content.

Trend archetype #2 grouped some strong international trends themed around society, containing Ellen DeGeneres' selfie picture taken at the Oscar ceremony as well as Robin Williams' sudden suicide. Archetype #3 contains trends with more specialized contents such as financial ("Facebook buys WhatsApp") or judicial ("Obergefell vs. Hodges, Supreme Court deciding on



(a) MH370 - March 8th 2014 *(b) Ferguson - Nov 25th 2014* *(c) MH370 - March 10th 2014*

(d) Ferguson - Nov 27th 2014 *(e) MH370 - March 12th 2014* *(f) Ferguson - Nov 29th 2014*

Figure 7.8: *Dissemination of trends "MH370" and "Ferguson"*

same-sex marriage").

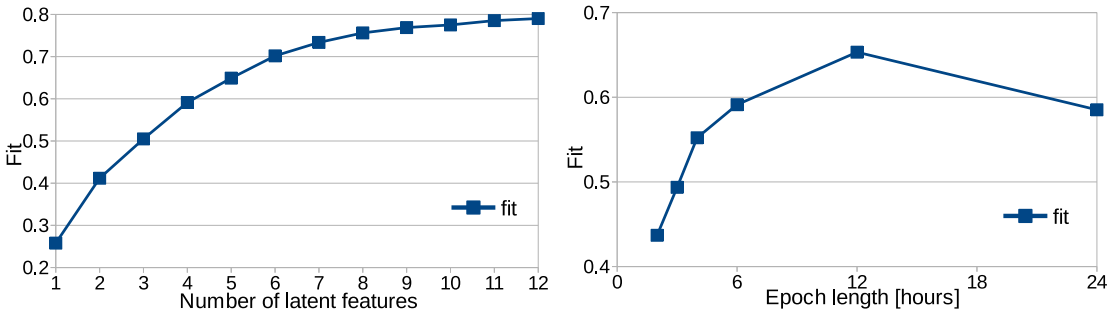
Another distinction is made between archetypes #4 and #5, both containing trends regarding the FIFA world cup 2014 in Brazil: while #4 represents game results and surprising or strong wins, #5 contains the more steady general discussion about the event, as well as other longer–term themes sparking much discussion. Among those is also the repeated outbreak of the Ebola virus in West Africa. Despite the entirely different nature of those topics, both represent a great public interest that dominated news media for longer periods of time.

7.2.5.3 Evaluation of approximation quality

The tensor decomposition employed in the flow modeling process exhibits a high quality for even low numbers of k, i.e., a small number of latent features per feature vector. This indicates large eigenvalues of the first k latent features, thus, indicating that these features are able to accurately describe the whole tensor with little loss of information. However, some information is still lost compared to an undecomposed tensor. The quality of the decomposition is evaluated by summing up the least-squared error between a reconstruction of the original tensor from its k-feature-vectors, and the original tensor itself. Let us refer to the inverse of this error as "fit", ranging from 1.0 for an exact match to 0.0 for no correlation.

Figure 7.9a shows that for a $k = 4$, the reconstructed tensor matches its original with a fit of 0.6, which is why $k = 4$ is chosen to be set in all subsequent experiments unless otherwise specified. As can be seen, the gain in fit slows down with additional latent features.

Figure 7.9b displays fit for different lengths of trend epochs, the granularity of the analysis in temporal dimension, ranging from 2 hours to 24 hours. The amount of days looked at per trend remained the same, so a longer epoch will result in a smaller number of epochs overall, reducing the size of $\mathscr{F}(\mathscr{D})$ in the T dimension. Intuitively, a smaller tensor $\mathscr{F}(\mathscr{D})$ is easier to reconstruct, increasing the fit for longer epochs. However, this does not hold for epochs of 24 hours. We



(a) Fit for number of latent features. *(b) Fit for length of trend episodes.*

Figure 7.9: *Approximation fit of factorized tensor.*

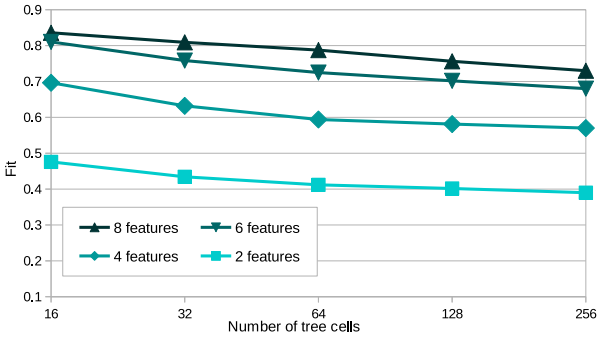


Figure 7.10: *Fit over tree cells for varying latent features.*

believe this to be due to a counter effect of more diversity in tree cell population as epochs get longer and, thus, more tweets are grouped in the same epoch. In other experiments, the epoch length was set to 6 hours unless otherwise specified – although it is not the peak for fit, we found it to best approximate trends from different global regions, hence, being able to compare trends in different hemispheres where peaks happen at different hours in the day.

The effect of varying spatial resolution can be seen in Figure 7.10 for four alternative settings of k. Although the underlying k-d tree is built on global tweet distribution to assure tweets in the same region from different trends are matched to the same cell, varying its node capacity upon indexing results in a higher- or lower resolved spatial grid, hence, lowering or increasing the size of $\mathscr{F}(\mathscr{D})$ in both spatial dimensions. Naturally, a smaller grid is easier to approximate with the same amount of latent features, yet the experiments show that features have a much higher impact on approximation quality than changing spatial resolution. As can be seen, fit values do not deteriorate much for higher numbers of grid cells.

The impact of different numbers of trends $\tau_{K,T}$ is stronger, particularly for smaller k. Figure 7.11 displays fit values for four alternative settings of k and the number of trends ranging from 20 to 100. As in previous experiments, fit decreases as the size of $\mathscr{F}(\mathscr{D})$ increases. However, for higher k the effect is drastically smaller, maintaining a good approximation quality at the cost of a higher complexity.

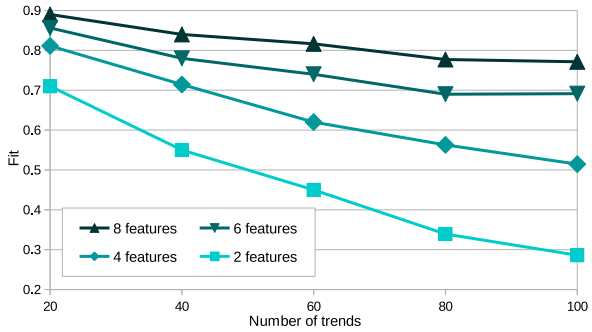


Figure 7.11: *Fit over trends for varying latent features.*

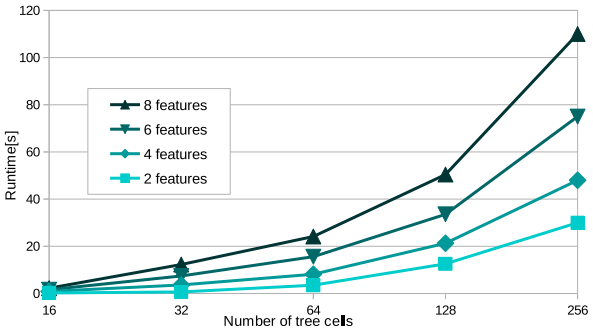


Figure 7.12: *Runtime over tree cells for varying latent features.*

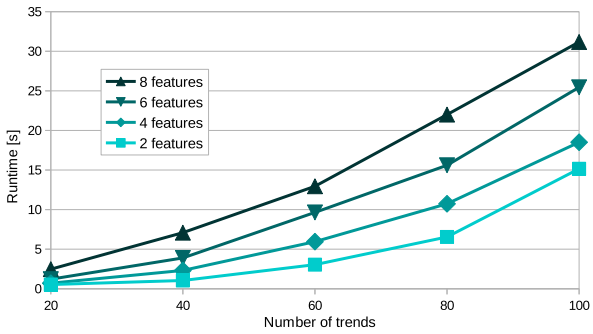


Figure 7.13: *Runtime over trends for varying latent features.*

7.2.5.4 Evaluation of algorithmic runtime

The following experiments evaluate runtime of the tensor generation, decomposition and projection on two modes $S \times S$. Filtering of tweets is not included in this evaluation since it depends heavily on the actual keyword settings as well as size of the underlying dataset. All experiments were performed on Arch Linux on an Intel i7 notebook with 16 GB of memory, implemented in python using numpy, pandas, and the sktensor package for tensor decomposition.

Figure 7.12 examines runtime in seconds over spatial resolution, for four different settings of k. Since an increase in the number of tree cells causes a quadradic increase in the size of $\mathscr{F}(\mathscr{D})$, runtimes scale superlinear for higher spatial resolutions.

The effect of different numbers of trends $\tau_{K,T}$ on runtime is shown in Figure 7.13. Runtimes show only a slight superlinear increase for higher amounts of trends, as the size of $\mathscr{F}(\mathscr{D})$ increases linearly with trends.

7.3 Summary

In this chapter, the thesis studied the dissemination of trends in space and time. For each historic trend, the proposed model constructs a spatio-temporal trend dissemination model, describing the flow of a trend through space and time. By applying a tensor factorization approach, latent features of trends are extracted, to which a clustering approach is applied to obtain sets of trends having a similar dissemination archetype. The qualitative evaluation of these trend archetypes on Twitter trends show meaningful dissemination archetypes, such as political trends, celebrity trends, and disaster trends. The quantitative analysis shows that the tensor factorization yields a high approximation quality for a low number of latent features. This result implies that a small number of latent features we derive from the flow of each trend is able to discriminate trends with a high-precision.

The next step of this research direction, is to make the trend flow based classification actionable for decision making. Thus, instead of classifying historic trends, the task is to deploy the system in an on-line streaming environment. For this purpose, the approach is to build a system which observes current and new trends (taken from existing trend mining solutions such as SigniTrend [185]), to classify the archetype of a trend as soon as possible, thus, allow to predict the spatio-temporal dissemination of trend. If successful, this approach will allow to predict the regional news of tomorrow, today.

8

Conclusion

I believe in intuition and inspiration. Imagination is more important than knowledge. For knowledge is limited, whereas imagination embraces the entire world, stimulating progress, giving birth to evolution. It is, strictly speaking, a real factor in scientific research.

Albert Einstein
1879-1955

Attribution

This Chapter uses material from all publications attributed in the previous Chapters.

See § 1.4 for an overview of incorporated publications and the author's attribution.

8.1 Summary & Conclusions

Complex graph systems describe relationships, dependencies, and the mutual exchange among individual data points reflecting real-world components. They encompass anything where multiple components interact with each other. Graph Theory provides the mathematical tools to model the requirements necessary in an interlinked world. A plethora of business problems are complex by nature and require graph thinking. Modern graph analysis and systems are the focus of this thesis.

Chapter 3: Homogeneous Graphs
§ 3 focuses on advanced analytics on homogeneous graphs. A novel spectral clustering approach called SCAR [88] is introduced, which improves robustness and efficiency simultaneously. The framework achieves this by replacing the eigendecomposition of the Laplacian with a Nyström approximation. Given input data, the first step is calculating a kNN graph. The key idea is to sample data points having the highest degree in the constructed graph. These points are referred to as landmark points that reflect the data's dense areas. A matrix decomposition is applied only on the subsample of the whole dataset. By integrating the Nyström extension, the eigenvectors of the whole matrix can be approximated. In an iterative process, the graph information is filtered such that only edges are retained, reflecting the true communities in the graph. In each iteration, noise edges are removed according to a scoring function measuring the distance in the eigenspace. The runtime complexity $\mathcal{O}(n^3)$ can be reduced to $\mathcal{O}(nm^2 + m^3)$, where m denotes the size of the subsample with $m \ll n$. The in-depth evaluation shows a significant improvement in robustness against noisy edges in the constructed similarity graph as well as robustness w.r.t jitter in the original data. Therefore, it tackles the two most difficult types of noise for clustering. Applying the Nyström method, the proposed model shows consistently low runtimes while returning highly competitive clustering qualities on real-world benchmark datasets. A potential research direction is to extend the proposed approach with feature weighting approaches that adapt the initial affinity matrix.

Chapter 4: Attributed Graphs
In § 4, the thesis dives into novel techniques that apply *Graph Neural Networks* (GNN) on attributed real-world graphs. Generally, deep learning models show promising results in a variety of downstream tasks, like representation learning for classification. This thesis lays the theoretical foundation for integrating an expert heuristic within transformer-based GNNs. Therefore, the framework provides the keystones for future works surpassing the successive thought of message-passing to develop even more powerful architectures in graph learning. By endowing graph neural networks with a set of experts, we gain a new level of expressiveness. The thesis shows how the most representative expert for learning a node's representation is selected. However, the routing layer introduced for this selection task is also combinable with several graph

learning models. In doing so, the learning procedure for a node can be affiliated with a specific model. Generally speaking, the experts differ in how to process subgraphs starting from a query node. The thesis introduces the terminology of *Graph **S**he**l**l **A**ttention* (SEA) [65] for referring to the nodes in an expert's receptive field. In terms of the over-smoothing problem, it is shown that the framework captures varying short- and long-term dependencies expressed via individual experts simultaneously. An empirical study on various real-world benchmark datasets shows highly competitive results while economizing a large number of parameters compared to other state-of-the-art models. For future work, the shell attention mechanism can be equipped either with different embedding methods being applied on the shells or with varying hyperparameter setting being individually defined for each shell.

Chapter 5: Probabilistic Graphs
By nature, the occurrence and effects of most real-world phenomena come with a notion of chance. The study of probability encompasses the formal description of the notion of uncertainty. § 5 studies a graph problem, where uncertainty is expressed in the information transmission of one node to another. In the language of Graph Theory, we can define a function which maps to each edge in a graph to a probability. The problem discussed in this thesis is centered around the maximization of the information flow in an uncertain graph given a budget of k communication edges. From a theoretical point of view, there are two *NP*-hard problems to solve. First, the computation of the expected information flow of a subgraph towards a designated source node, and second, the selection of the optimal k-set of edges. In order to solve these problems, this thesis proposes a novel data structure, called the *Flow-Tree* (F-Tree) [61, 60, 63]. From a high level perspective, it decomposes a graph into several components which can be categorized either to be a *mono-connected component* - for which the information flow can be calculated analytically - or to be a *bi-connected component* - for which an Monte-Carlo sampling is applied in order to estimate the information flow. Speaking in terms of *Graph Theory*, the former type of component relates to tree structures in a graph, whereas the second type relates to component, where at least one cycle can be observed in the extracted subgraph. From a practical point of view, the F-Tree is especially suitable for graphs where a locality assumptions holds such as for location-aware/spatial networks (e.g., road networks).

Chapter 6: Spatial Graphs
§ 6 introduces a novel variant of the vehicle routing problem (VRP; cf. Toth and Vigo [202]). Specifically, the problem is extended in two dimensions. First, the scheduling considers time windows being defined for each customer individually. Second, the problem implies flexible capacitated delivery locations, i.e., a customer's service is flexible w.r.t the location a customer is served. In the first phase, an initial solution is constructed where a novel backtracking mechanism is applied. Greedily, customers are inserted in the scheduling. If the resulting solution is infeasible, the procedure jumps back to one of the former steps and explores the solution space by sus-

pending insertions, having been observed leading to infeasibilities. After the construction step, an improvement phase starts. In order to solve the optimization problem, a metaheuristic based on the *Adaptive Large Neighborhood Search* (ALNS; cf. [173]) is proposed, a well-established framework for solving routing problems. The framework is extended by a *Guided Local Search* (GLS). The extended ALNS framework iteratively destroys parts of this solution using randomly selected destroy operators and reconstructs the destroyed solution with randomly selected repair operators [66]. The combination of destroy and repair operators defines the neighborhood where the new solution will be sought. The procedure allows for temporary infeasibilities in solutions to further enhance the exploration of the solution space. The Guided Local Search dynamically adjusts how these infeasibilities are penalized. The algorithm is evaluated on a set of generic instances designed to represent different hospital layouts and various demand scenarios. The extension with GLS and novel repair and destroy procedures show superior performance to the standard ALNS. Moreover, the evaluation gives insights into how different cost functions for assigning customers to locations different from the preferred location affect the overall planning. Noteworthy, a more in-depth analysis of the interplay between hyperparameters steering the ALNS and the parameters steering the GLS is likely to lead to a more robust version of the novel algorithm and presents a promising research direction.

Chapter 7: Social Media Graphs

A subfield studying social systems using graph-theoretic tools is known as *Social Network Analysis* (SNA). In § 7, social analysis is applied to examine the dissemination of trends in both dimensions, space and time. The proposed model is called TrendTracker [183]. The links in the underlying graph reflect the observations of trends occurring in connected locations within a specific time interval. Hence, the vertices in the graph are the locations with labels on linkages denoting the trend being observed in a particular time interval. This yields adjacency matrices reflecting the trend flows between areas. Considering all temporal time intervals, we get a 3-dimensional tensor and, finally, a 4-dimensional tensor when considering additionally all topics. A tensor factorization is then applied to reveal the latent features. Finally, a clustering approach is applied to obtain sets of trends having a similar dissemination archetype. The evaluation shows that trends referring to political topics, celebrities, or disasters have unique dissemination patterns. A promising research direction is to apply the approach of TrendTracker on so-called exchange networks, where dissemination pattern between various entities have to be examined from the spatial and temporal domain.

8.2 Outlook & Perspectives of Graph AI

I want to conclude this thesis with a personal outlook of Graph AI from two angles: **i)** what are open research questions (cf. § 8.2.1) and **ii)** the relevance of network science from a business' value point of view (cf. § 8.2.2).

Generally speaking, explainable AI, interpretability, transparency, and trustworthiness will play pivotal roles in the success story of Graph AI. The disclosure of the mystery of *'black boxes'* is a predominant argument for the success of deep learning in practical applications where rule-based algorithms are still favorable. The integration of meta-backed *Knowledge Graphs* will help in cases of imbalanced training datasets or in cases where data is of poor quality. Graph Theory and its implementation in products and platforms will help transition data into knowledge. The value of graph applications gives incentives to invest in AI techniques in light of a competitive economy being also induced by these techniques. The integration of graph-theoretic analysis will help businesses to make a big step forward in our connected world.

8.2.1 Scientific Outlook

Explainable Graph AI.
In order to be implementable in the decision-making process of business applications, predictions made by deep learning systems must be interpretable by experts and, probably even more importantly, explainable to the public. Explainable AI provides a human-friendly way to analyze semantically linked data enabling visualization of decision paths computed by AI models. This implies, for example, analyzing the context of where the information came from, which methods were used, what meta-data is being used, where is the data coming from being used, is there bias in the data, and so forth. Therefore, explainability is a required component of AI to reach acceptance in more sectors, and being context driven by graph-backed algorithms improves explainability.

Dynamic Graph AI.
Living in a constantly changing environment, we must adapt our choices according to dynamically varying circumstances. In 2018, it was shown that methods for static knowledge graphs are not always feasible for the case of *temporal Knowledge Graphs* (tKGs), like in the case of Knowledge Graph completion [127, 69, 36]. Thus, new methods are needed for graph-theoretic analysis to analyze temporal data evolving over the temporal axis. Dynamic graph learning algorithms are also of particular interest in traffic networks, in social network analysis to identify concept drifts in the data or in various streaming environments.

Generative and Self-supervised Graph AI.
Whereas in recent years, the research community provided sophisticated generative models for constructing graphs fulfilling homophily properties, a heterophily property brings up new challenges. Inspired by *Generative Adversarial Networks* (GANs), generative graph learning models unify a generative and discriminative component playing a game-theoretical min-max game. The ability to distinguish between artificial graphs and real-world graphs has shown promising results in downstream tasks like link prediction, recommendation systems, and the analysis of network evolution. In this sense, it is also noteworthy that self-supervised learning (SSL) will play a crucial role in AI losing the necessity of a full-labeled training dataset that resembles the human learning process to a greater extent.

Combinatorial Optimization with Graph AI.
Combinatorial optimization problems are the family of integer constrained optimization problems that are NP-hard. Traditionally, classical optimization problems such as *Traveling Salesman Problem* (TSP), *Job-shop Scheduling Problem* (JSP), or *Vehicle Routing Problem* (VRP; cf. [202]) have been solved by exact algorithms which work up to a limited amount of nodes. Heuristics and metaheuristics have been established for larger problem instances like § 6 presents it for the VRPTW-FL. Recent end-to-end approaches leverage advances in graph representation learning and have shown competitive performance with OR solvers. Bengio et al. [19] illustrates how machine learning in general can be applied in combinatorial optimization. The interdisciplinary research of Neural combinatorial optimization shows the synergies between DL and CO algorithms merging the best of two domains to develop new algorithms, especially for applied problems.

Meta-data Graph AI.
The semantic analysis of graphs encompasses vertical as well as horizontal analysis, which will further help in understanding processes in practical applications. A vertical analysis includes the identification of categories in a domain, whereas a horizontal analysis is of interest in cross-cutting concerns. New and open research questions arise when we want to link various taxonomies and ontologies to enable interoperability. In a world of an increasing amount of data, new ways have to be developed to integrate data from multiple information sources to provide a de-duplicated view of the data one is interested in. Meta-data layers have shown their benefits in practical applications, where organization-wide data is connected such that a unified view of local graph data with data retrieved from third-party systems is provided. The value of such an ecosystem provides not only insights into the data lineage but also cross-functional information.

Overarching/Multimodal Graph AI.
Remarkably, Machine Learning and Deep Learning models significantly improve their decision process when contextual information is taken into account. A specific issue herewith is graph modeling allows AI architectures to look across a number of different datasets to infer context. The increased information provided by the relationships between various data points increase the information an AI model is fed with which result in more accurate and ultimately in more human-like, cognitive conclusions. This enables an AI system to learn if anomalies occur only local or not.

8.2.2 Practical Relevance

In gaining better insights into modern deep learning architectures and formal Graph Theory in general, an increasing number of platforms and applications leverage network science for sophisticated data analysis. Nowadays, high-performance computing allows for processing graphs containing millions and billions of entities. For example, the latest note about the size of Google's Knowledge Graph in mid-2020: *"It [Google's Knowledge Graph; author's note] has amassed over 500 billion facts about five billion entities."* [28]. Therefore, data analytics enhanced with graph theory gets pervasive all over various business branches. Microsoft Azure's Cosmos DB for Apache Gremlin[29], AWS's Neptune [30], Neo4j[31], or TigerGraph[32] are examples of how accelerating and adopting graph analytics in modern data analytics with all the confidence and alacrity. Deep Learning architectures implementing graph neural networks in combination with technologies and tools becoming affordable for a broad range of end users is one of the most promising and brightest trends in enterprise IT. The evolution of open-source libraries for modern AI technologies, frameworks, high-performant engines, and specialized accelerators leads to solutions with high-end AI models to predict developments across various industry verticals in an automated way. Graph Theory provides the tools and algorithms to connect the thinkers of tomorrow today.

[28] https://blog.google/products/search/about-knowledge-graph-and-knowledge-panels/; last access Oct 18, 2022
[29] https://learn.microsoft.com/en-us/azure/cosmos-db/gremlin/introduction
[30] https://aws.amazon.com/de/neptune/
[31] https://neo4j.com/
[32] https://www.tigergraph.com/

References

[1] Emmanuel Abbe. Community detection and stochastic block models. *Found. Trends Commun. Inf. Theory*, June 2018. ISSN 1567-2190. doi: 10.1561/0100000067. URL `https://doi.org/10.1561/0100000067`.

[2] Eytan Adar and Christopher Ré. Managing uncertainty in social networks. *IEEE Data Eng. Bull.*, 30(2):15–22, July 2007.

[3] KK Aggarwal, KB Misra, and JS Gupta. Reliability evaluation a comparative study of different techniques. *Microelectronics Reliability*, 14(1):49–56, 1975.

[4] Annalisa Appice, Anna Ciampi, and Donato Malerba. Summarizing numeric spatial data streams by trend cluster discovery. *Data Mining and Knowledge Discovery*, 29(1):84–136, 2015.

[5] Walter Edwin Arnoldi. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of applied mathematics*, 9(1):17–29, 1951.

[6] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[7] Christian Artigues, Philippe Michelon, and Stéphane Reusser. Insertion techniques for static and dynamic resource-constrained project scheduling. *European Journal of Operational Research*, 149(2):249 – 267, 2003. ISSN 0377-2217.

[8] Audi AG. Audi, dhl and amazon deliver convenience, 2015. `https://www.audi-mediacenter.com/en/press-releases/audi-dhl-and-amazon-deliver-convenience-347`.

[9] Nabila Azi, Michel Gendreau, and Jean-Yves Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. *Comput. Oper. Res.*, 41:167–173, January 2014. ISSN 0305-0548.

[10] Francis Bach and Michael Jordan. Learning spectral clustering. *Advances in neural information processing systems*, 16(2):305–312, 2004.

[11] Sivaraman Balakrishnan, Min Xu, Akshay Krishnamurthy, and Aarti Singh. Noise thresholds for spectral clustering. *Advances in Neural Information Processing Systems*, 24, 2011.

[12] R. Baldacci, E. Bartolini, and G. Laporte. Some applications of the generalized vehicle routing problem. *Journal of the Operational Research Society*, 61(7):1072–1077, 2010. ISSN 1476-9360.

[13] Maria Battarra, Jean-François Cordeau, and Manuel Iori. *Chapter 6: Pickup-and-Delivery Problems for Goods Transportation*, pages 161–191. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[14] Dominique Beaini, Saro Passaro, Vincent Létourneau, William L. Hamilton, Gabriele Corso, and Pietro Liò. Directional graph networks. *CoRR*, 2020. URL `https://arxiv.org/abs/2010.02863`.

[15] J. E. Beasley and E. M. Nascimento. The vehicle routing-allocation problem: A unifying framework. *TOP*, 4(1):65–86, 1996. ISSN 1863-8279.

[16] Anna Beer, Ekaterina Allerborn, Valentin Hartmann, and Thomas Seidl. Kiss-a fast knn-based importance score for subspaces. In *EDBT*, pages 391–396, 2021.

[17] Tolga Bektaş, Güneş Erdoğan, and Stefan Røpke. Formulations and branch-and-cut algorithms for the generalized vehicle routing problem. *Transportation Science*, 45(3):299–316, 2011.

[18] Serge Belongie, Charless Fowlkes, Fan Chung, and Jitendra Malik. Spectral partitioning with indefinite kernels using the nyström extension. In *European conference on computer vision*, pages 531–542. Springer, 2002.

[19] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2020.07.063. URL `https://www.sciencedirect.com/science/article/pii/S0377221720306895`.

[20] Aleksandar Bojchevski, Yves Matkovic, and Stephan Günnemann. Robust spectral clustering for noisy data: Modeling sparse corruptions improves latent embeddings. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 737–746. ACM, 2017.

[21] Anne-Laure Boulesteix. Ten simple rules for reducing overoptimistic reporting in methodological computational research. *PLoS Computational Biology*, 11(4):e1004191, 2015.

[22] Timothy B Brecht and Charles J Colbourn. Lower bounds on two-terminal network reliability. *Discrete Applied Mathematics*, 21(3):185–198, 1988.

[23] Peter Brucker, Andreas Drexl, Rolf Möhring, Klaus Neumann, and Erwin Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112(3):3 – 41, 1999.

[24] M. Bruglieri, S. Mancini, and O. Pisacane. The green vehicle routing problem with capacitated alternative fuel stations. *Computers & Operations Research*, 112:104759, 2019. ISSN 0305-0548.

[25] Dov Bulka and Joanne Bechta Dugan. Network st reliability bounds using a 2-dimensional reliability polynomial. *Reliability, IEEE Transactions on*, 43(1):39–45, 1994.

[26] Eamonn Cahill, Alan Irving, Christopher Johnston, James Sexton, Ukqcd Collaboration, et al. Numerical stability of lanczos methods. *Nuclear Physics B-Proceedings Supplements*, 83:825–827, 2000.

[27] J. Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970. ISSN 1860-0980. doi: 10.1007/BF02310791. URL `http://dx.doi.org/10.1007/BF02310791`.

[28] Arthur Cayley. On the theory of the analytical forms called trees. *Philosophical Magazine*, 13:172–6, 1857.

[29] Jr. Cecil C. Bridges. Hierarchical cluster analysis. *Psychological Reports*, 18(3):851–854, 1966. doi: 10.2466/pr0.1966.18.3.851. URL `https://doi.org/10.2466/pr0.1966.18.3.851`.

[30] Xiaojun Chen, Weijun Hong, Feiping Nie, Dan He, Min Yang, and Joshua Zhexue Huang. Spectral clustering of large-scale data by directly solving normalized cut. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1206–1215, 2018.

[31] Anna Choromanska, Tony Jebara, Hyungtae Kim, Mahesh Mohan, and Claire Monteleoni. Fast spectral clustering via the nyström method. In *International Conference on Algorithmic Learning Theory*, pages 367–381. Springer, 2013.

[32] Charles J Colbourn and CJ Colbourn. *The combinatorics of network reliability*, volume 200. Oxford University Press New York, 1987.

[33] J-F Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52(8): 928–936, Aug 2001. ISSN 1476-9360.

[34] J-F Cordeau, M. Gendreau, G. Laporte, J-Y Potvin, and F. Semet. A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, 53(5):512–522, 2002. ISSN 1476-9360.

[35] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005. doi: 10.1016/j.jalgor.2003.12.001.

[36] Shib Sankar Dasgupta, Swayambhu Nath Ray, and Partha Talukdar. HyTE: Hyperplane-based temporally aware knowledge graph embedding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2001–2011, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[37] William A. Dees, Jr. and Patrick G. Karger. Automated rip-up and reroute techniques. In *Proceedings of the 19th Design Automation Conference*, DAC '82, pages 432–439, Piscataway, NJ, USA, 1982. IEEE Press. ISBN 0-89791-020-6.

[38] Guy Desaulniers, Oli B.G. Madsen, and Stefan Ropke. *Chapter 5: The Vehicle Routing Problem with Time Windows*, pages 119–159. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[39] M. Desrochers, J.K. Lenstra, and M.W.P. Savelsbergh. A classification scheme for vehicle routing and scheduling problems. *European Journal of Operational Research*, 46(3):322–332, 1990. ISSN 0377-2217.

[40] M. Desrochers, C.V. Jones, J.K. Lenstra, M.W.P. Savelsbergh, and L. Stougie. Towards a model and algorithm management system for vehicle routing and scheduling problems. *Decision Support Systems*, 25(2):109 – 133, 1999. ISSN 0167-9236.

[41] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556, 2004.

[42] Karl F. Doerner and Juan-José Salazar-González. *Chapter 7: Pickup-and-Delivery Problems for People Transportation*, pages 193–212. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[43] Pedro Domingos and Matt Richardson. Mining the network value of customers. In *SIGKDD*, pages 57–66, 2001.

[44] Michael Drexl. Synchronization in vehicle routing–a survey of VRPs with multiple synchronization constraints. *Transportation Science*, 46(3):297–316, 2012.

[45] Petros Drineas and Michael W Mahoney. Approximating a gram matrix for improved kernel-based learning. In *International Conference on Computational Learning Theory*, pages 323–337. Springer, 2005.

[46] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, New York, 2 edition, 2001. ISBN 978-0-471-05669-0.

[47] Dorian Dumez, Fabien Lehuédé, and Olivier Péton. A large neighborhood search approach to the vehicle routing problem with delivery options. *Transportation Research Part B: Methodological*, 144:103 – 132, 2021.

[48] Dorian Dumez, Christian Tilk, Stefan Irnich, Fabien Lehuédé, and Olivier Péton. Hybridizing large neighborhood search and exact methods for generalized vehicle routing problems with time windows (revision 2). *EURO Journal on Transportation and Logistics*, page 100040, 2021. ISSN 2192-4376.

[49] Vijay Prakash Dwivedi and Xavier Bresson. A generalization of transformer networks to graphs, 2021.

[50] Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

[51] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, 57(4):1472–1483, 2009. ISSN 0360-8352.

[52] Tobias Emrich, Hans-Peter Kriegel, Johannes Niedermayer, Matthias Renz, André Suhartha, and Andreas Züfle. Exploration of monte-carlo based probabilistic query processing in uncertain graphs. In *CIKM*, pages 2728–2730, 2012.

[53] P. Erdös and A Rényi. On the evolution of random graphs. In *Publication of the mathematical institute of the hungarian academy of sciences*, pages 17–61, 1960.

[54] Thomas Ericsson and Axel Ruhe. The spectral transformation lanczos method for the numerical solution of large sparse generalized symmetric eigenvalue problems. *Mathematics of Computation*, 35(152):1251–1268, 1980.

[55] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[56] Leonhard Euler. Solutio problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140, 1736.

[57] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *CoRR*, 2021. URL https://arxiv.org/abs/2101.03961.

[58] T. Finch. Incremental calculation of weighted mean and variance. Technical report, University of Cambridge, 2009.

[59] Charless Fowlkes, Serge Belongie, Fan Chung, and Jitendra Malik. Spectral grouping using the nyström method. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), 2004.

[60] Christian Frey, Andreas Züfle, Tobias Emrich, and Renz. Efficient information flow maximization in probabilistic graphs (extended abstract). In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1801–1802. IEEE Computer Society, 2018. doi: 10.1109/ICDE.2018.00258. URL https://doi.org/10.1109/ICDE.2018.00258.

[61] Christian Frey, Andreas Züfle, Tobias Emrich, and Matthias Renz. Efficient information flow maximization in probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30:880–894, 2018.

[62] Christian M. M. Frey and Matthias Schubert. V-coder: Adaptive autoencoder for semantic disclosure in knowledge graphs, 2022. URL https://arxiv.org/abs/2208.01735.

[63] Christian M. M. Frey, Andreas Züfle, Tobias Emrich, Matthias Renz, and Regine Meunier. Efficient data propagation in a computer network. https://pubchem.ncbi.nlm.nih.gov/patent/US-2020394249-A1, 2016. Accessed: 2022-9-24.

[64] Christian M. M. Frey, Yunpu Ma, and Matthias Schubert. Apptek: Agent-based predicate prediction in temporal knowledge graphs. *CoRR*, abs/2110.14284, 2021.

[65] Christian M. M. Frey, Yunpu Ma, and Matthias Schubert. Sea: Graph shell attention in graph neural networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022.

[66] Christian M.M. Frey, Alexander Jungwirth, Markus Frey, and Rainer Kolisch. The vehicle routing problem with time windows and flexible delivery locations. *European Journal of Operational Research*, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.11.029. URL https://www.sciencedirect.com/science/article/pii/S0377221722008888.

[67] Aurélien Froger, Ola Jabali, Jorge E. Mendoza, and Gilbert Laporte. The electric vehicle routing problem with capacitated charging stations. *Transportation Science*, 56(2):460–482, 2022.

[68] J Galtier, A Laugier, and P Pons. Algorithms to evaluate the reliability of a network. In *DRCN*, pages 8–pp, 2005.

[69] Alberto García-Durán, Sebastijan Dumančić, and Mathias Niepert. Learning sequence encoders for temporal knowledge graph completion. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4816–4821, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[70] Daniel Gartner, Markus Frey, and Rainer Kolisch. Hospital-wide therapist scheduling and routing: Exact and heuristic methods. *IISE Transactions on Healthcare Systems Engineering*, 8(4):268–279, 2018.

[71] Gianpaolo Ghiani and Gennaro Improta. An efficient transformation of the generalized vehicle routing problem. *European Journal of Operational Research*, 122(1):11 – 17, 2000. ISSN 0377-2217. doi: https://doi.org/10.1016/S0377-2217(99)00073-9. URL `http://www.sciencedirect.com/science/article/pii/S0377221799000739`.

[72] Joy Ghosh, Hung Q Ngo, Seokhoon Yoon, and Chunming Qiao. On a routing problem within probabilistic graphs and its application to intermittently connected networks. In *INFOCOM*, pages 1721–1729, 2007.

[73] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL `https://proceedings.mlr.press/v70/gilmer17a.html`.

[74] Alex Gittens and Michael Mahoney. Revisiting the nystrom method for improved large-scale machine learning. In *International Conference on Machine Learning*, pages 567–575. PMLR, 2013.

[75] Fred Glover and Jin-Kao Hao. The case for strategic oscillation. *Annals of Operations Research*, 183(1):163–173, 2011. ISSN 1572-9338.

[76] B. Golden, S. Raghavan, and E. A. Wasil, editors. *The vehicle routing problem: Latest Advances and New Challenges*. Springer US, Boston, MA, 2008.

[77] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *WWW*, pages 403–412, 2004.

[78] Stephan Günnemann, Ines Färber, Sebastian Raubach, and Thomas Seidl. Spectral subspace clustering for graphs with feature vectors. In *2013 IEEE 13th International Conference on Data Mining*, pages 231–240. IEEE, 2013.

[79] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2017. URL https://proceedings.neurips.cc/paper/2017/file/5dd9db5e033da9c6fb5ba83c7a7ebea9-Paper.pdf.

[80] Gary Hardy, Corinne Lucet, and Nikolaos Limnios. K-terminal network reliability measures with binary decision diagrams. *Reliability, IEEE Transactions on*, 56(3):506–515, 2007.

[81] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an" explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16(1): 84, 1970.

[82] Ville Hautamaki, Ismo Karkkainen, and Pasi Franti. Outlier detection using k-nearest neighbour graph. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 430–433. IEEE, 2004.

[83] Bruce Hendrickson and Robert W. Leland. A multi-level algorithm for partitioning graphs. *Supercomputing '95: Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (CDROM)*, pages 1–14, 1995.

[84] Sibylle Hess, Wouter Duivesteijn, Philipp Honysz, and Katharina Morik. The spectacl of nonconvex clustering: a spectral approach to density-based clustering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3788–3795, 2019. ISBN 2374-3468.

[85] Desmond J Higham and Milla Kibble. A unified view of spectral clustering. *University of Strathclyde mathematics research report*, 2, 2004.

[86] Petteri Hintsanen. *The Most Reliable Subgraph Problem*, pages 471–478. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74976-9. doi: 10.1007/978-3-540-74976-9_48. URL https://doi.org/10.1007/978-3-540-74976-9_48.

[87] Sin C. Ho, W.Y. Szeto, Yong-Hong Kuo, Janny M.Y. Leung, Matthew Petering, and Terence W.H. Tou. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological*, 111:395–421, 2018. ISSN 0191-2615.

[88] Ellen Hohma∗, Christian M. M. Frey∗, Anna Beer∗, and Thomas Seidl. Scar: Spectral clustering accelerated and robustified. *Proc. VLDB Endow.*, 15(11):3031–3044, jul 2022. ISSN 2150-8097. doi: 10.14778/3551793.3551850. URL https://doi.org/10.14778/3551793.3551850.

[89] Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bull. Amer. Math. Soc.*, August 2006. ISSN 0273-0979. doi: 10.1090/s0273-0979-06-01126-8.

[90] John Hopcroft and Robert Tarjan. Algorithm 447: efficient algorithms for graph manipulation. *Communications of the ACM*, 16(6):372–378, 1973.

[91] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *arXiv preprint arXiv:2005.00687*, 2020.

[92] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference 2020*, WWW '20, 2020. ISBN 9781450370233. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027.

[93] Ming Hua and Jian Pei. Probabilistic path queries in road networks: traffic uncertainty aware path selection. In *EDBT*, 2010.

[94] Hao Huang, Shinjae Yoo, Hong Qin, and Dantong Yu. A robust clustering algorithm based on aggregated heat kernel mapping. In *2011 IEEE 11th International Conference on Data Mining*, pages 270–279. IEEE, 2011. ISBN 1457720752.

[95] Ling Huang, Donghui Yan, Nina Taft, and Michael Jordan. Spectral clustering with perturbed data. *Advances in Neural Information Processing Systems*, 21, 2008.

[96] Tülin Inkaya. A parameter-free similarity graph for spectral clustering. *Expert Syst. Appl.*, 42(24):9489–9498, dec 2016. ISSN 0957-4174. doi: 10.1016/j.eswa.2015.07.074. URL https://doi.org/10.1016/j.eswa.2015.07.074.

[97] Stefan Irnich, Paolo Toth, and Daniele Vigo. *Chapter 1: The Family of Vehicle Routing Problems*, chapter The Family of Vehicle Routing Problems, pages 1–33. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[98] John J Irwin, Teague Sterling, Michael M Mysinger, Erin S Bolstad, and Ryan G Coleman. Zinc: A free tool to discover chemistry for biology. *Journal of chemical information and modeling*, 2012. ISSN 1549-9596.

[99] Anil K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31 (8):651 – 666, 2010. ISSN 0167-8655.

[100] Hongjie Jia, Shifei Ding, Hong Zhu, Fulin Wu, and Lina Bao. A feature weighted spectral clustering algorithm based on knowledge entropy. *J. Softw.*, 8(5):1101–1108, 2013.

[101] Ruoming Jin, Lin Liu, Bolin Ding, and Haixun Wang. Distance-constraint reachability computation in uncertain graphs. *Proceedings of the VLDB Endowment*, 4(9):551–562, 2011.

[102] A. Jungwirth, M. Frey, and R. Kolisch. The vehicle routing problem with time windows, flexible service locations and time-dependent location capacity. Technical report, Technical University of Munich, 2020.

[103] Alexander Jungwirth, Guy Desaulniers, Markus Frey, and Rainer Kolisch. Exact branch-price-and-cut for a hospital therapist scheduling problem with flexible service locations and time-dependent location capacity. *INFORMS Journal on Computing*, 34(2):1157–1175, 2022.

[104] Janani Kalyanam, Amin Mantrach, Diego Saez-Trumper, Hossein Vahabi, and Gert Lanckriet. Leveraging social context for modeling topic evolution. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 517–526. ACM, 2015.

[105] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[106] Melissa Kasari, Hannu Toivonen, and Petteri Hintsanen. Fast discovery of reliable k-terminal subgraphs. In Mohammed Javeed Zaki, Jeffrey Xu Yu, B. Ravindran, and Vikram Pudi, editors, *PAKDD*, volume 6119, pages 168–177, 2010. ISBN 978-3-642-13671-9. URL `http://dblp.uni-trier.de/db/conf/pakdd/pakdd2010-2.html#KasariTH10`.

[107] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Introduction to NP-Completeness of knapsack problems*. Springer, 2004.

[108] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *SIGKDD*, pages 137–146, 2003.

[109] Arijit Khan, Francesco Bonchi, Aristides Gionis, and Francesco Gullo. Fast reliability search in uncertain graphs. In *EDBT*, pages 535–546, 2014.

[110] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[111] Thomas N. Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. In *Proceedings of the 5th International Conference on Learning Representations*, ICLR '17, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

[112] G. Kirchhoff. Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird, January 1847. URL https://doi.org/10.1002/andp.18471481202.

[113] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983. ISSN 0036-8075.

[114] George Kollios, Michalis Potamias, and Evimaria Terzi. Clustering large probabilistic graphs. *Knowledge and Data Engineering, IEEE Transactions on*, 25(2):325–336, 2013.

[115] Oumar Koné, Christian Artigues, Pierre Lopez, and Marcel Mongeau. Event-based milp models for resource-constrained project scheduling problems. *Computers & Operations Research*, 38(1):3–13, 2011. Project Management and Scheduling.

[116] Attila A. Kovacs, Sophie N. Parragh, and Richard F. Hartl. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. *Networks*, 63(1): 60–81, 2014. ISSN 1097-0037.

[117] Devin Kreuzer, Dominique Beaini, William L. Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention, 2021.

[118] Joseph B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. ISSN 00029939, 10886826.

[119] Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling techniques for the nystrom method. In *Artificial Intelligence and Statistics*, pages 304–311, 2009.

[120] Rahma Lahyani, Mahdi Khemakhem, and Frédéric Semet. Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1):1 – 14, 2015. ISSN 0377-2217.

[121] Edward Lam and Pascal Van Hentenryck. A branch-and-price-and-check model for the vehicle routing problem with location congestion. *Constraints*, 21:394 – 412, 2016. ISSN 1572-9354.

[122] Cornelius Lanczos. *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators*. United States Governm. Press Office Los Angeles, CA, 1950.

[123] Gilbert Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, November 2009. ISSN 1526-5447.

[124] Gilbert Laporte and Ibrahim H. Osman. Routing problems: A bibliography. *Annals of Operations Research*, 61(1):227–262, 1995. ISSN 1572-9338.

[125] Gilbert Laporte, Stefan Ropke, and Thibaut Vidal. *Chapter 4: Heuristics for the vehicle routing problem*, pages 87–116. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[126] Theodoros Lappas, Marcos R Vieira, Dimitrios Gunopulos, and Vassilis J Tsotras. On the spatiotemporal burstiness of terms. *Proceedings of the VLDB Endowment*, 5(9):836–847, 2012.

[127] Julien Leblay and Melisachew Wudage Chekol. Deriving validity time in knowledge graph. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, page 1771–1776, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404.

[128] Daniel Lehmann, Liadan Ita Oćallaghan, and Yoav Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM (JACM)*, 49(5):577–602, 2002.

[129] Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

[130] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *SIGKDD'06*, pages 631–636, 2006.

[131] Jure Leskovec and Julian J Mcauley. Learning to discover social circles in ego networks. In *NIPS*, pages 539–547, 2012.

[132] Jurij Leskovec, Deepayan Chakrabarti, Jon Kleinberg, and Christos Faloutsos. Realistic, mathematically tractable graph generation and evolution, using kronecker multiplication. In *ECML-PKDD*, pages 133–145. Springer, 2005.

[133] Jianzhong Li, Zhaonian Zou, and Hong Gao. Mining frequent subgraphs over uncertain graph databases under probabilistic semantics. *The VLDB Journal*, 21(6):753–777, 2012.

[134] Mu Li, James Tin-Yau Kwok, and Baoliang Lu. Making large-scale nyström approximation possible. In *ICML 2010-Proceedings, 27th International Conference on Machine Learning*, page 631, 2010.

[135] Mu Li, Xiao-Chen Lian, James T Kwok, and Bao-Liang Lu. Time and space efficient spectral clustering via column sampling. In *CVPR 2011*, pages 2297–2304. IEEE, 2011. ISBN 1457703955.

[136] Xi Li, Weiming Hu, Chunhua Shen, Anthony Dick, and Zhongfei Zhang. Context-aware hypergraph construction for robust spectral clustering. *IEEE Transactions on Knowledge and Data Engineering*, 26(10):2588–2597, 2014. doi: 10.1109/TKDE.2013.126.

[137] Yuan Li, Haoxun Chen, and Christian Prins. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. *European Journal of Operational Research*, 252(1):27 – 38, 2016. ISSN 0377-2217.

[138] Zhenguo Li, Jianzhuang Liu, Shifeng Chen, and Xiaoou Tang. Noise robust spectral clustering. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. ISBN 1424416302.

[139] Cindy Xide Lin, Bo Zhao, Qiaozhu Mei, and Jiawei Han. Pet: a statistical model for popular events tracking in social communities. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 929–938. ACM, 2010.

[140] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[141] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[142] Simona Mancini. A real-life multi depot multi period vehicle routing problem with a heterogeneous fleet: Formulation and adaptive large neighborhood search based matheuristic. *Transportation Research Part C: Emerging Technologies*, 70:100–112, 2016. ISSN 0968-090X.

[143] Vittorio Maniezzo, Marco Antonio Boschetti, and Thomas Stützle. *Automatic Design for Matheuristics*, pages 35–57. Springer International Publishing, Cham, 2021. ISBN 978-3-030-70277-9.

[144] F. E. Maranzana. On the location of supply points to minimize transport costs. *OR*, 15(3): 261–270, 1964. ISSN 14732858.

[145] Renaud Masson, Fabien Lehuéd é, and Olivier Péton. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, 47(3): 344–355, 2013.

[146] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. Measurement and Analysis of Online Social Networks. In *Proceedings of*

the 5th ACM/Usenix Internet Measurement Conference (IMC'07), San Diego, CA, October 2007.

[147] L. Moccia, J-F Cordeau, and G. Laporte. An incremental tabu search heuristic for the generalized vehicle routing problem with time windows. *Journal of the Operational Research Society*, 63(2):232–244, 2012. ISSN 1476-9360.

[148] Mahesh Mohan and Claire Monteleoni. Exploiting sparsity to improve the accuracy of nyström-based large-scale spectral clustering. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 9–16. IEEE, 2017. ISBN 1509061827.

[149] Prakash Nadkarni. Chapter 10 - core technologies: Data mining and "big data". In Prakash Nadkarni, editor, *Clinical Research Computing*, pages 187–204. Academic Press, 2016. ISBN 978-0-12-803130-8. doi: https://doi.org/10.1016/B978-0-12-803130-8.00010-5. URL https://www.sciencedirect.com/science/article/pii/B9780128031308000105.

[150] W. Navidi. *Statistics for Engineers and Scientists*. McGraw-Hill Higher education. McGraw-Hill, 2006. ISBN 9780071214926. URL https://books.google.de/books?id=jiCOAAAACAAJ.

[151] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14:849–856, 2001.

[152] Dai Quoc Nguyen, Tu Dinh Nguyen, and Dinh Phung. Universal self-attention network for graph classification. *arXiv preprint arXiv:1909.11855*, 2019.

[153] Kenta Oono and Taiji Suzuki. Graph neural networks exponentially lose expressive power for node classification. *arXiv: Learning*, 2020.

[154] Gizem Ozbaygin, Oya Ekin Karasan, Martin Savelsbergh, and Hande Yaman. A branch-and-price algorithm for the vehicle routing problem with roaming delivery locations. *Transportation Research Part B: Methodological*, 100:115 – 137, 2017. ISSN 0191-2615.

[155] Odysseas Papapetrou, Ekaterini Ioannou, and Dimitrios Skoutas. Efficient discovery of frequent subgraph patterns in uncertain graph databases. In *EDBT*, pages 355–366, 2011.

[156] Beresford N. Parlett. *The Symmetric Eigenvalue Problem*. Society for Industrial and Applied Mathematics, Philadelphia, 1998. ISBN 0898714028.

[157] Sophie N. Parragh and Jean-François Cordeau. Branch-and-price and adaptive large neighborhood search for the truck and trailer routing problem with time windows. *Computers & Operations Research*, 83:28 – 44, 2017. ISSN 0305-0548.

[158] Philippa Pattison and Garry Robins. *Handbook of Probability: Theory and Applications*, pages 291–310. SAGE Publications, Inc., 0 edition, 2008. doi: http://dx.doi.org/10.4135/9781452226620. URL `http://dx.doi.org/10.4135/9781452226620`.

[159] Jella Pfeiffer and Franz Rothlauf. Analysis of greedy heuristics and weight-coded eas for multidimensional knapsack problems and multi-unit combinatorial auctions. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation*, pages 1529–1529, 2007.

[160] David Pisinger and Stefan Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34(8):2403–2435, 2007. ISSN 0305-0548.

[161] David Pisinger and Stefan Ropke. *Large Neighborhood Search*, pages 99–127. Springer US, Boston, MA, 3 edition, 2019. ISBN 978-3-319-91086-4.

[162] Michalis Potamias, Francesco Bonchi, Aristides Gionis, and George Kollios. k-nearest neighbors in uncertain graphs. *PVLDB*, 3(1):997–1008, 2010. URL `http://dblp.uni-trier.de/db/journals/pvldb/pvldb3.html#PotamiasBGK10`.

[163] Jean-Yves Potvin and Jean-Marc Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

[164] Farhad Pourkamali-Anaraki. Scalable spectral clustering with nyström approximation: Practical and theoretical aspects. *IEEE Open Journal of Signal Processing*, 1:242–256, 2020.

[165] Caroline Prodhon and Christian Prins. A survey of recent research on location-routing problems. *European Journal of Operational Research*, 238(1):1 – 17, 2014. ISSN 0377-2217.

[166] J Scott Provan and Michael O Ball. Computing network reliability in time polynomial in the number of cuts. *Operations Research*, 32(3):516–526, 1984.

[167] Damián Reyes, Martin Savelsbergh, and Alejandro Toriello. Vehicle routing with roaming delivery locations. *Transportation Research Part C: Emerging Technologies*, 80:71 – 91, 2017. ISSN 0968-090X.

[168] Douglas A. Reynolds. Gaussian mixture models. In Stan Z. Li and Anil K. Jain, editors, *Encyclopedia of Biometrics*, pages 659–663. Springer US, 2009. ISBN 978-0-387-73003-5.

[169] Matthew Richardson and Pedro Domingos. Mining knowledge-sharing sites for viral marketing. In *SIGKDD*, pages 61–70, 2002.

[170] Alan Ritter, Oren Etzioni, Sam Clark, et al. Open domain event extraction from twitter. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1104–1112. ACM, 2012.

[171] Simone Romano, Nguyen Xuan Vinh, James Bailey, and Karin Verspoor. Adjusting for chance clustering comparison measures. *The Journal of Machine Learning Research*, 17 (1):4635–4666, 2016.

[172] Yu Rong, Yatao Bian, Tingyang Xu, Weiyang Xie, Ying WEI, Wenbing Huang, and Junzhou Huang. Self-supervised graph transformer on large-scale molecular data. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, 2020. URL `https://proceedings.neurips.cc/paper/2020/file/94aef38441efa3380a3bed3faf1f9d5d-Paper.pdf`.

[173] Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[174] Stefan Ropke and David Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–775, 2006. ISSN 0377-2217. Feature Cluster: Heuristic and Stochastic Methods in Optimization Feature Cluster: New Opportunities for Operations Research.

[175] Gerardo Rubino. Network performance modeling and simulation. In Jean Walrand, Kallol Bagchi, and George W. Zobrist, editors, *Network Performance Modeling and Simulation*, chapter Network Reliability Evaluation, pages 275–302. Gordon and Breach Science Publishers, Inc., Newark, NJ, USA, 1999. ISBN 90-5699-596-0. URL `http://dl.acm.org/citation.cfm?id=308004.308025`.

[176] Tomoya Sakai and Atsushi Imiya. Fast spectral clustering with random projection and sampling. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 372–384. Springer, 2009.

[177] Takeshi Sakaki, Makoto Okazaki, and Yutaka Matsuo. Earthquake shakes twitter users: real-time event detection by social sensors. In *Proceedings of the 19th international conference on World wide web*, pages 851–860. ACM, 2010.

[178] Jagan Sankaranarayanan, Hanan Samet, Benjamin E Teitler, Michael D Lieberman, and Jon Sperling. Twitterstand: news in tweets. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 42–51. ACM, 2009.

[179] Paolo Santi. *Topology Control in Wireless Ad Hoc and Sensor Networks*. John Wiley & Sons, New York, 2005.

[180] Martin Savelsbergh and Tom Van Woensel. 50th anniversary invited article - city logistics: Challenges and opportunities. *Transportation Science*, 50(2):579–590, 2016.

[181] Hassan Sayyadi, Matthew Hurst, and Alexey Maykov. Event detection and tracking in social streams. In *ICWSM*, 2009.

[182] Maximilian Schiffer and Grit Walther. An adaptive large neighborhood search for the location-routing problem with intra-route facilities. *Transportation Science*, 52(2):331–352, 2018.

[183] Klaus Arthur Schmid, Christian Frey, Fengchao Peng, Michael Weiler, Andreas Züfle, Lei Chen, and Matthias Renz. Trendtracker: Modelling the motion of trends in space and time. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1145–1152. IEEE Computer Society, 2016. doi: 10.1109/ICDMW.2016.0165. URL `https://doi.org/10.1109/ICDMW.2016.0165`.

[184] Gerhard Schrimpf, Johannes Schneider, Hermann Stamm-Wilbrandt, and Gunter Dueck. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, 159(2):139 – 171, 2000. ISSN 0021-9991.

[185] Erich Schubert, Michael Weiler, and Hans-Peter Kriegel. Signitrend: scalable detection of emerging topics in textual streams by hashed significance thresholds. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 871–880. ACM, 2014.

[186] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, November 2020. ISSN 0001-0782. doi: 10.1145/3381831. URL `https://doi.org/10.1145/3381831`.

[187] Petteri Sevon, Lauri Eronen, Petteri Hintsanen, Kimmo Kulovesi, and Hannu Toivonen. Link discovery in graphs derived from biological databases. In *DILS*, pages 35–49, 2006.

[188] Ahmad R Sharafat and Omid Reza Ma'rouzi. All-terminal network reliability using recursive truncation algorithm. *Reliability, IEEE Transactions on*, 58(2):338–347, 2009.

[189] Paul Shaw. *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*, pages 417–431. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-49481-2.

[190] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[191] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. ISSN 0162-8828.

[192] C. Sinoquet and R. Mourad. *Probabilistic graphical models for genetics, genomics, and postgenomics*, chapter xxx. Oxford University Press, 2014.

[193] Katayoun Sohrabi, Jay Gao, Vishal Ailawadhi, and Gregory J Pottie. Protocols for self-organization of a wireless sensor network. *IEEE personal communications*, 7(5):16–27, 2000.

[194] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–265, 1987.

[195] G. W. Stewart. A krylov-schur algorithm for large eigenproblems. *SIAM J. Matrix Anal. Appl.*, 23(3):601–614, 2002. doi: 10.1137/S0895479800371529.

[196] Alexander Strehl and Joydeep Ghosh. Cluster ensembles – a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research (JMLR)*, 3: 583–617, December 2002. ISSN 1533-7928.

[197] Thomas Stützle and Manuel López-Ibáñez. *Automated Design of Metaheuristic Algorithms*, pages 541–579. Springer International Publishing, Cham, 2019. ISBN 978-3-319-91086-4.

[198] Zhiqiang Tao, Hongfu Liu, Sheng Li, and Yun Fu. Robust spectral ensemble clustering. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 367–376, 2016.

[199] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972.

[200] Christian Tilk, Katharina Olkis, and Stefan Irnich. The last-mile vehicle routing problem with delivery options. *OR Spectrum*, 2021.

[201] Joshua Tobin and Mimi Zhang. Dcf: An efficient and robust density-based clustering method. In *2021 IEEE International Conference on Data Mining (ICDM)*, pages 629–638. IEEE, 2021.

[202] P. Toth and D. Vigo. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2002.

[203] P. Toth and D. Vigo. *Vehicle Routing*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2014.

[204] Nicolas Tremblay and Andreas Loukas. Approximating spectral clustering via sampling: a review. *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 129–183, 2020.

[205] Sayan Unankard, Xue Li, and Mohamed A Sharaf. Emerging event detection in social networks with location sensitivity. *World Wide Web*, 18(5):1393–1417, 2015.

[206] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 2017. ISBN 9781510860964.

[207] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations*, 2017.

[208] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, 40(1):475 – 489, 2013. ISSN 0305-0548.

[209] Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, and Christian Prins. Time-window relaxations in vehicle routing heuristics. *Journal of Heuristics*, 21(3):329–358, 2015. ISSN 1572-9397.

[210] Thibaut Vidal, Gilbert Laporte, and Piotr Matl. A concise guide to existing and emerging vehicle routing problem variants. *European Journal of Operational Research*, 286(2):401 – 416, 2020. ISSN 0377-2217.

[211] Edwin von Boventer. The relationship between transportation costs and location rent in transportation problems. *Journal of Regional Science*, 3(2):27–40, 1961. ISSN 1467-9787.

[212] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007. ISSN 0960-3174.

[213] Christos Voudouris and Edward Tsang. Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, 113(2):469 – 499, 1999. ISSN 0377-2217.

[214] Liang Wang, Christopher Leckie, Kotagiri Ramamohanarao, and James Bezdek. Approximate spectral clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Advances in Knowledge Discovery and Data Mining, pages 134–146. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-01307-2.

[215] Minjie Wang, Da Zheng, Zihao Ye, Quan Gan, Mufei Li, Xiang Song, Jinjing Zhou, Chao Ma, Lingfan Yu, Yu Gai, Tianjun Xiao, Tong He, George Karypis, Jinyang Li, and Zheng Zhang. Deep graph library: A graph-centric, highly-performant package for graph neural networks. *arXiv preprint arXiv:1909.01315*, 2019.

[216] CDT Watson-Gandy and PJ Dohrn. Depot location with van salesmen - a practical approach. *Omega*, 1(3):321 – 329, 1973. ISSN 0305-0483.

[217] M. H. J. Webb. Cost functions in the location of depots for multiple-delivery journeys. *Journal of the Operational Research Society*, 19(3):311–320, 1968. ISSN 1476-9360.

[218] B. P. Welford. Note on a method for calculating corrected sums of squares and products. *Technometrics*, 4(3):419–420, 1962. doi: 10.2307/1266577.

[219] Barry Wellman, Anabel Quan Haase, James Witte, and Keith Hampton. Does the internet increase, decrease, or supplement social capital?: Social networks, participation, and community commitment. *American Behavioral Scientist*, 45(3):436–455, 2001. doi: 10.1177/00027640121957286. URL `https://doi.org/10.1177/00027640121957286`.

[220] Jianshu Weng and Bu-Sung Lee. Event detection in twitter. *ICWSM*, 11:401–408, 2011.

[221] D. H. D. West. Updating mean and variance estimates: an improved method. *CommunicationsACM*, 22(9):532–535, 1979. doi: 10.1145/359146.359153.

[222] Jeffery Westbrook and Robert E Tarjan. Maintaining bridge-connected and biconnected components on-line. *Algorithmica*, 7(1):433–464, 1992.

[223] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. In *Advances in neural information processing systems*, pages 682–688, 2001.

[224] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2021. doi: 10.1109/TNNLS.2020.2978386.

[225] He Xu, Lin Zhang, Peng Li, and Feng Zhu. Outlier detection algorithm based on k-nearest neighbors-local outlier factor. *Journal of Algorithms & Computational Technology*, 16: 17483026221078111, 2022. doi: 10.1177/17483026221078111. URL https://doi.org/10.1177/17483026221078111.

[226] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *CoRR*, 2018. URL http://arxiv.org/abs/1810.00826.

[227] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *ICDM*. IEEE Computer Society, 2012.

[228] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform bad for graph representation? *arXiv preprint arXiv:2106.05234*, 2021.

[229] Ye Yuan, Guoren Wang, Lei Chen, and Haixun Wang. Efficient keyword search on uncertain graph data. *TKDE*, 25(12):2767–2779, 2013.

[230] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J Kim. Graph transformer networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, 2019. URL https://proceedings.neurips.cc/paper/2019/file/9d63484abb477c97640154d40595a3bb-Paper.pdf.

[231] Kai Zhang, Liang Lan, Jun Liu, Andreas Rauber, and Fabian Moerchen. Inductive kernel low-rank decomposition with priors: A generalized nystrom method. *arXiv preprint arXiv:1206.4619*, 2012.

[232] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. *A Data-Driven Graph Generative Model for Temporal Interaction Networks*, page 401–411. Association for Computing Machinery, 2020. ISBN 9781450379984. URL https://doi.org/10.1145/3394486.3403082.

[233] Xiangmin Zhou and Lei Chen. Event detection over twitter social media streams. *The VLDB journal*, 23(3):381–400, 2014.

[234] Xiatian Zhu, Chen Change Loy, and Shaogang Gong. Constructing robust affinity graphs for spectral clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1450–1457, 2014.

[235] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Finding top-k maximal cliques in an uncertain graph. In *ICDE*, pages 649–652, 2010.

[236] Zhaonian Zou, Jianzhong Li, Hong Gao, and Shuo Zhang. Mining frequent subgraph patterns from uncertain graph data. *Knowledge and Data Engineering, IEEE Transactions on*, 22(9):1203–1218, 2010.

Own Publications

Main Publications

[61] Christian Frey, Andreas Züfle, Tobias Emrich, and Matthias Renz. Efficient information flow maximization in probabilistic graphs. *IEEE Transactions on Knowledge and Data Engineering*, 30:880–894, 2018.

[60] Christian Frey, Andreas Züfle, Tobias Emrich, and Renz. Efficient information flow maximization in probabilistic graphs (extended abstract). In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 1801–1802. IEEE Computer Society, 2018. doi: 10.1109/ICDE.2018.00258. URL https://doi.org/10.1109/ICDE.2018.00258.

[63] Christian M. M. Frey, Andreas Züfle, Tobias Emrich, Matthias Renz, and Regine Meunier. Efficient data propagation in a computer network. https://pubchem.ncbi.nlm.nih.gov/patent/US-2020394249-A1, 2016. Accessed: 2022-9-24.

[65] Christian M. M. Frey, Yunpu Ma, and Matthias Schubert. Sea: Graph shell attention in graph neural networks. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 2022.

[88] Ellen Hohma∗, Christian M. M. Frey∗, Anna Beer∗, and Thomas Seidl. Scar: Spectral clustering accelerated and robustified. *Proc. VLDB Endow.*, 15(11): 3031–3044, jul 2022. ISSN 2150-8097. doi: 10.14778/3551793.3551850. URL https://doi.org/10.14778/3551793.3551850.

[66] Christian M.M. Frey, Alexander Jungwirth, Markus Frey, and Rainer Kolisch. The vehicle routing problem with time windows and flexible delivery locations. *European Journal of Operational Research*, 2022. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2022.11.029. URL https://www.sciencedirect.com/science/article/pii/S0377221722008888.

[183] Klaus Arthur Schmid, Christian Frey, Fengchao Peng, Michael Weiler, Andreas Züfle, Lei Chen, and Matthias Renz. Trendtracker: Modelling the motion of trends in space and time. In Carlotta Domeniconi, Francesco Gullo, Francesco Bonchi, Josep Domingo-Ferrer, Ricardo Baeza-Yates, Zhi-Hua Zhou, and Xindong Wu, editors, *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, pages 1145–1152. IEEE Computer Society, 2016. doi: 10.1109/ICDMW.2016.0165. URL https://doi.org/10.1109/ICDMW.2016.0165.

Further Publications (ongoing)

[62] Christian M. M. Frey and Matthias Schubert. V-coder: Adaptive autoencoder for semantic disclosure in knowledge graphs, 2022. URL `https://arxiv.org/abs/2208.01735`.

[64] Christian M. M. Frey, Yunpu Ma, and Matthias Schubert. Apptek: Agent-based predicate prediction in temporal knowledge graphs. *CoRR*, abs/2110.14284, 2021.

List of Figures

List of Tables