Navigation with uncertain Spatio-temporal Resources

Dissertation von Sebastian Schmoll

München 2021

Navigation with uncertain Spatio-temporal Resources

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

vorgelegt/eingereicht von
Sebastian Schmoll

aus
München

14.10.2020

Erstgutachter/in:	Prof. Dr. Matthias Schubert
Zweitgutachter/in:	Prof. Dr. Herke van Hoof
Drittgutachter/in:	Prof. Dr. Goce Trajcevski
Tag der mündlichen Prüfung:	04.02.2021

Eidesstattliche Versicherung

Hiermit erkläre ich, Sebastian Schmoll, an Eides statt, dass die vorliegende Dissertation ohne unerlaubte Hilfe gemäß Promotionsordnung vom 12.07.2011, § 8, Abs. 2 Pkt. 5, angefertigt worden ist.

München, 07.04.2021

Sebastian Schmoll

Contents

Zusammenfassung

Durch intelligente Navigationssysteme werden Verkehrsteilnehmer davor bewahrt, Umwege zu fahren. Dadurch sparen sie Zeit, Geld und verringern den CO_2-Ausstoß. Aus diesem Grund verbauen Hersteller Navigationssysteme in fast allen Neuwägen. Bis heute unterstützen die meisten Systeme nur einfache Routenplanung, die den kürzesten oder schnellsten Pfad von A nach B berechnen. Dennoch müssen Fahrer regelmäßig Entscheidungen darüber hinaus treffen. Beispielsweise soll eine möglichst günstige Tankstelle auf dem Weg zum eigentlichen Ziel besucht werden. Allerdings kann diese ihre Preise, während der Fahrer oder die Fahrerin auf dem Weg dort hin ist, dynamisch ändern. Anschließend muss, sobald das eigentliche Ziel erreicht ist, ein Parkplatz gefunden werden. Bisher fahren Parkplatzsuchende zufällig durch das Zielgebiet in der Hoffnung möglichst schnell einen freien Parkplatz zu finden. Die Suche verursacht zusätzlichen Verkehr und der Fahrer oder die Fahrerin verbringt mehr Zeit auf der Straße. Neben Privatpersonen müssen auch Transportunternehmen komplexe Entscheidungen über Bewegungen treffen. Zum Beispiel muss ein Taxifahrer, wenn er gerade keinen Fahrgast hat, entscheiden, wo er sich als nächstes positioniert. Zwar könnte er am letzten Zielort warten, bis er einen Anruf der Taxizentrale bekommt. Falls jedoch der letzte Zielort in einem entlegenen Gebiet ist, muss der nächste Fahrgast wahrscheinlich lange warten, bis der Fahrer oder die Fahrerin bei ihm ankommt. Damit sinkt die Kundenzufriedenheit, was wiederum einen potentiellen Verlust der Kunden bedeutet. Seit Kurzem gibt es immer mehr Datenquellen, die Entscheidungen für diese Probleme verbessern. Beispielsweise wird durch Parkplatzsensoren die Verfügbarkeit der Parkplätze verfolgt, mobile Anwendungen sammeln Anfragen über Fahrgäste und Tankstellen veröffentlichen ihren aktuellen Preis in Echtzeit. In dieser Arbeit wird der Forschungsfrage nachgegangen, wie Algorithmen gestaltet werden können, sodass diese veränderlichen Informationen verwendet werden können. Standard-Routing-Algorithmen gehen von einer statischen Welt aus. Aber die Verfügbarkeit von Fahrgästen, die Tankstellenpreise und die Parkplatzzustände ändern sich nicht deterministisch. Aus diesem Grund modellieren wir eine Reihe von Anwendungen als Markov-Entscheidungsproblem (MDP). Applikationsabhängig schlagen wir vor, das MDP mit dynamischer Programmierung, Replanning bzw. Hindsight Planning oder Reinforcement Learning zu lösen. Abschließend fassen wir alle Anwendungen in einer Domäne zusammen. Dadurch können wir einen Reinforcement Learning Ansatz definieren, der alle Anwendungen in dieser Domäne ohne Änderung lösen kann. Dieser Ansatz ermöglicht es, die Routenplanung von der eigentlichen Problemstellung zu lösen. Dadurch ist die gelernte Funktionsapproximation auch auf bisher unbekannte Straßennetze ohne weiteres Training anwendbar.

Abstract

Supporting people with intelligent navigation instructions enables users to efficiently achieve trip-related objectives (e.g., minimum travel time or fuel consumption) and preserves them from making unnecessary detours. This, in turn, enables them to save time, money and, additionally, minimize CO_2 emissions. For these reasons, manufacturers integrate navigation systems into almost all modern automobiles. Nevertheless, most of them support only simple routing instructions, i.e., how to drive from location A to B. Albeit, people are regularly faced with more complex decisions, e.g. navigating to a cheap gas station on the route while incorporating dynamic gas price changes. Another example-scenario is after reaching the destination, an available facility to park needs to be found. So far, people cruise almost randomly around the goal area in the search for a parking space. As a consequence, persons valuable time is consumed and unnecessary traffic arises. Besides private persons, transportation companies have to make complex mobility decisions. For instance, taxi drivers have to find out where to move next whenever the taxi is idle. There are plenty possibilities for where the taxi driver could go. In case the last drop-off was in a sparsely populated region, waiting for a call from the taxi office will likely result in a longer drive to the next customer. In turn, customer satisfaction decreases with a longer waiting time and implies a potential loss of customers.

Recently, the number of data sources that potentially improve these mobility decisions increased. For instance, on-street parking sensors track the current state of the spaces (e.g. Melbourne), mobile applications collect taxi requests from customers and gas stations publish the current prices all in real-time. This thesis investigates the question of how to design algorithms such that they exploit this volatile data. Standard routing algorithms assume a static world. But the availability of passengers, gas prices and the availability of parking spots change over time in a non-deterministic manner. Hence, we model multiple real-world applications as Markov decision processes (MDP), i.e., a framework for sequential decision making under uncertainty. Depending on the task, we propose to solve the MDP with dynamic programming, replanning and hindsight planning or reinforcement learning. Ultimately, we combine all applications in a single problem domain. Subsequently, we propose a reinforcement learning approach that solves all applications in this domain without modification. Furthermore, it decouples the routing task from solving the application itself. Hence, it is transferable to previously unseen street networks without further training.

Chapter 1

Introduction

The geographic mobility of persons and goods dramatically increased within the last century. In modern civilization, people perform different tasks in various places, such that the locations for working, housing and leisure activities are scattered. Hence, people are constantly on the move. Furthermore, goods such as groceries or other consumer items are delivered around the globe. However, the fluctuation in modern civilization has its price. In Germany, the transport sector emitted 20 percent of the CO_2 emissions (as of May 2020), which corresponds to 163 million tons of greenhouse gases [38]. Those gases reinforce climate change and this, among others, increases the number and strength of natural disasters (wildfire, flooding, hurricane). Furthermore, the higher sea-level reduces valuable soil, which in turn decreases living environment and food production. As the climate change overall has appalling consequences, the goal is to minimize the CO_2 emissions. The research community and industry are working on alternatives to combustion engines. Nevertheless, mobility will always consume energy and if we want to exclusively use renewable energies, we have to keep in mind the limited volume that renewable sources provide.

In developed countries, traveling is an integral part of daily life. There are two main logistic categories – public and individual transportation (and sometimes a particular trip may combine the two models). From an energy consumption point of view, public transportation is much more efficient, since it bundles a bunch of people or goods in a single truck, or train. Specifically, in the case of trains, a frequently electric engine moves hundreds of individuals or goods along the same way. It further causes less particulate matter in cities. Nonetheless, the bundling has its drawbacks. The start and destination of the public transport system are fixed and not equal to the actual routes of the different individuals. Hence, they require another type of mobility for the so called last mile, i.e., the first/last portion of the respective trips. The carrying of heavy or large items is less comfortable. Economically oriented organizations often decide in favor of individual transportation, since lower travel times save personnel costs and customer satisfaction can often be increased. Moreover, as the recent COVID-19 pandemic has demonstrated, the epidemic spread is facilitated by crowded train compartments. The more or less frequent trips and sometimes not duly schedule of trains and buses restrict the planning capabilities of the individuals. The energy consumption benefits in sparse regions is put into perspective by fewer individuals to share the commute. In any case, private transportation is more flexible and time-efficient than public traffic connection, particularly but not limited in non-urban regions.

Individual transportation gives people a high degree of flexibility. As a simple example, at every turning point, a car or truck driver can decide which direction to choose, i.e., a driver can move from any point in a street network to another. But doing so requires a sequence of decisions, i.e., at every upcoming intersection they want to pick the direction that leads them to their destination. Finding the shortest path from a location A to another location B is called routing and is mostly considered to be solved in static environments. For decades, navigation systems have been relieving drivers of this sequential decision-making task. Nonetheless, mobility is more than routing. For instance, an employee wants to drive home from the office, but he/she wants to stop by a (cheap) gas/charging station and an ATM along the way. The car needs a place to park on-street close to the densely populated residence. The availability of parking spots is volatile. Hence, the shortest path solutions fails to provide an effective method for encompassing all the requirements of the mobility-related task. Let us consider a further example. Suppose we are in a smart city or amusement park that collects information about the occupancy and waiting times of attractions. People want to visit a set of places. If we do not consider waiting times, and reduce the walking times only for a given bulk of attractions, this setting becomes an instance of the travelling salesman problem. For a pleasant stay, people want to visit various types of point of interests (restaurants, sights, rides) alternately. Thereby, they want to increase the number of visits, ranked by personal preferences, within a given time-period. Since the status of the attractions changes in a non-deterministic manner, a pre-computed plan for the whole day (path) is not adequate. Most research so far focused on static environments. However, due to the increasing amount of available data-sources, taking into consideration the stochastic future changes of potential interesting locations can further improve the solutions, implying shorter on-street time and less CO_2 emissions.

Such potentially interesting locations with stochastic development occur in many real-world applications. We summarize them by the definition of stochastic spatial resources (SSR). Other illustrative examples are parking spots, taxi passengers or rental bicycles. They all share the property of having a location and status that changes over time according to a stochastic process. People may directly alter the state of the SSRs, e.g. occupying the parking spot or picking up a taxi passenger.

Here is where our research comes in. This thesis tackles the question of how to support people with navigation decisions by using data about SSRs provided by smart cities or mobile applications. This is a sequential decision making task under uncertainty.

We present several applications in which an agent navigates along a road graph and visits SSRs. One such task is the parking search problem, where the agent searches an available parking spot close to the destination. We model this problem as an instance of a Markov decision process (MDP) and solve it optimally with dynamic programming (DP) techniques. Since the state space and by implication the computational efforts grow exponentially with the number of parking spots, DP is only suitable for small problem settings. Even after only updating relevant parts of the state space and optimizing the transition function, the computational efforts become intractable with an increasing number of locations that are subject to updates. Thus, for scalability, we seek good approximate solutions. Some real-world applications are not “probabilistic interesting”, which means that choosing a non-optimal option likely results in a significantly worse situation. For instance, picking a sub-optimal turn during a parking search is typically reversible with little additional costs. For this reason, we can show that replanning and hindsight planning approaches work well within the parking search task. Nevertheless, in some cases, even the determinization is once again an instance of the travelling salesman problem and accordingly NP-hard [25].

Even with existing data for a particular task, defining the MDP (particularly the state transition probabilities) may be difficult or error-prone. Therefore, we provide a broad framework based on deep reinforcement learning that applies to the issues described above without modification and the need for an MDP model.

This thesis is structured as follows. Chapter 2 gives an in-depth overview of this thesis research area, previous research and applications. Afterwards, Chapter 3 sketches the contributions of this work. Then, we conclude and recommend directions for future research.

Chapter 2

Mobility Tasks with Stochastic Spatial Resources

This chapter highlights the previous research gap filled by this thesis's research. We explain the term stochastic spatial resources (SSR), illustrate the research gap which this research closes and list example applications treated within this work.

2.1 Stochastic Spatial Resources

It often requires movement to get a particular task done. For example, people desire to visit a specific target location (e.g. office) or type of target (e.g. restaurant). Generally, these places have volatile states, e.g. an ATM can be closed, a gas station has changing prices, and a restaurant may be fully occupied. We call such targets stochastic spatial resources (SSR). As mentioned, the problem settings addressed in this thesis assume that all the resources have a location. While this is static for some types of resources (e.g. ATM, gas station), the resource's place may change over time for other instances, e.g. an ice cream cart moves over time. An SSR has a potentially volatile state, e.g. availabilities, prices, etc. We assume that a stochastic process changes both SSR properties over time. Some resources may be altered directly or indirectly by the agent. For instance, a guest (agent) occupies a restaurant table, or a thief takes flight from a policeman (agent).

Hitherto, real-time information about SSRs was sparse. However, due to sensors and mobile applications, the observability about the states and locations becomes increasingly accessible. Future applications will expectedly consider this information to provide their customers with better navigation decisions.

2.2 Classification of the Research Area

Below, we provide a positioning of the research area in the context of existing classifications of spatial and artificial intelligence (AI) research, illustrated in Figure 2.1.

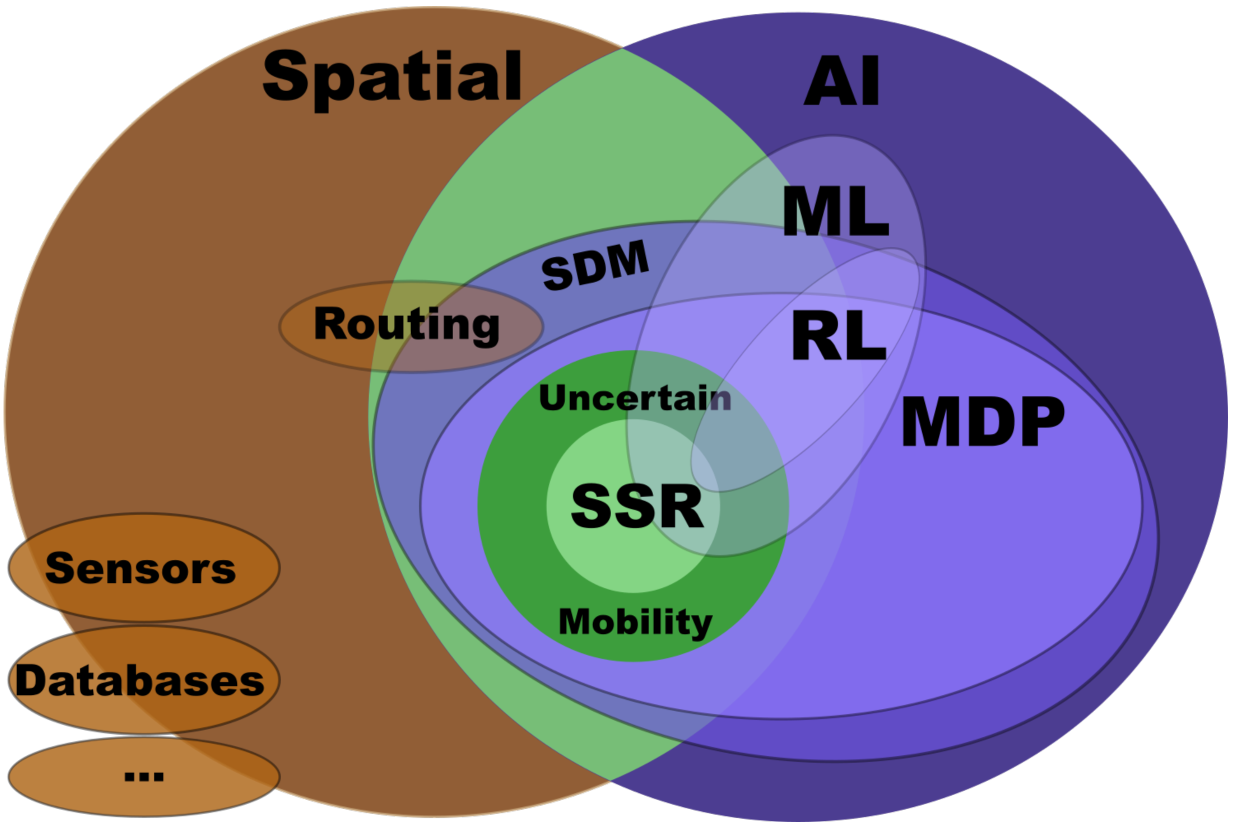


Figure 2.1: Placement of the SSR research within the spatial and AI research. The illustration contains multiple research areas (circles) that are enclosed within others or intersect. The ratio of a subset of the super-set in the illustration does not indicate the real portion.

2.2.1 Spatial

Improving today's mobility is a core topic of the spatial research community. State-of-the-art navigation systems gather data from numerous sources (e.g. sensor data or mobile applications) to enhance navigation decision to efficiently guide users to a specific destination. Hence, the spatial community combines knowledge gathered from diverse other research fields. For instance, the spatial community integrates research on database and data mining technologies among others, because data has to be efficiently stored, retrieved and processed. Due to crowd sources and sensor value errors, the data is regularly uncertain [35, 11]. Uncertainty due to incomplete information about the state is hereinafter called partially observability [54]. Even if we certainly know the real current state (i.e., fully observable), the future development usually follows a probability distribution. Therefore, the spatial community uses and adapts machine learning tools from the AI community to learn the expected future state (e.g. regression) [10, 6, 30] or to sample from the distribution (e.g. generative adversarial networks) [40]. Those machine learning models can then be helpful for human decision making.

2.2.2 Sequential Decision Making

Making a sequence of decisions to reach a specific goal is called sequential decision making (SDM). It has a sustained research background in artificial intelligence (AI). In some applications the transition model is deterministic, i.e., a given state-action pair always implies the same next state. Then, search algorithms find the solution (here a sequence of actions) for a given start state. In some applications, people are happy to find any solution. In others, they want to minimize the number of actions (bread-first search) or the costs for executing the sequence like, for example minimizing the travel distance from a given source to a given destination (using Dijkstra's algorithm [7]). Search algorithms differ in the order in which they expand state-action pairs. The goal is to find a solution with as few expansions as possible. For this purpose, informed search algorithms use heuristics, e.g. A* [14]. However, many heuristics are problem-specific. Nevertheless, applications are plentiful, e.g. sliding puzzles, routing, robot navigation, automatic assembly sequencing and protein design. Consequently, research on application-independent heuristics is available, e.g. FF [15]. (cf. [37], Chapter 3 and 10)

2.2.3 Routing

Only in the 20th century, cartography became more widely available to both individuals and companies, and its effective use was enabled due to print techniques and areal- and satellite images. Prior to that, little help was available for deciding how to find (an efficient route to) a specific location. Access to automobiles tremendously increased the mobility range of the large portion of the population, and the demand for routing to previously unknown areas has become more frequent. It was common to look up and memorize an easy route rather than the shortest or fastest route. At every turning point, it was necessary to make a decision based on the previous preparation. Especially in complex paths, partial information about the current position made decisions while driving error-prone. Today, GPS and regularly updated map data simplify the decisions drives have to do on-street, as the navigation system tells at each intersection which direction to take next. Modern map-systems are even aware of current traffic congestion. Consequently, they save the driver time and reduce CO_2 emissions.

Although the first systems were able to reduce stress and preparation and driving time, path computations for long distances were time consuming and difficult due to low computational power. Spatial research tackled this issue. Researchers proposed better lower bounds compared to the areal-distance, e.g. landmarks [13]. Hierarchical approaches (e.g. highway hierarchies [39]) split the task into two different action abstractions, such that long-distance routing is efficiently done on a reduced, higher-level graph (highways).

2.2.4 Uncertainty

So far, we assumed a deterministic world. However, future developments are stochastic in some applications. We call such tasks sequential decision making under uncertainty. A framework for such problems is the Markov decision process (MDP).

Since the follow-state is probabilistic, searching for a goal state is not useful. We do not know if the assumed state sequence will occur. Further, we cannot make any guarantee about the costs (or rewards) received in a particular trial. Hence, we optimize the expected costs or rewards in an episode. An MDP describes a probability distribution for the next state given the current state and action. Accordingly, MDPs describe a Markovian (memoryless) stochastic process. In [45], we argue why MDPs are well-fitting to uncertain mobility tasks. Methodologies for solving MDPs already exist in general-purpose AI literature, e.g. value iteration [3], policy iteration [16] and q-learning [52]. However, they do not converge in an adequate amount of time in realistic settings. In the last decade, developments of the graphics processing unit (GPU) technology and deep learning frameworks utilizing GPUs lead to advantages in function approximation with artificial neural networks. With approximations (e.g. deep neural networks), the algorithm can assign close representations, a similar utility and abstract experiences between states. By virtue of these advances, deep reinforcement learning has reached the capability to solve much larger problems.

The ambition of general-purpose AI is to develop algorithms that maximize the number of environments for which an effective solution can be generated. Nonetheless, for the specific applications, it makes sense to exploit knowledge about the problem (e.g. spatio-temporal knowledge) to let it converge in a reasonable time. Utilizing this knowledge has been done before in deterministic contexts, i.e., routing. The deterministic heuristic search in general-purpose AI (e.g. FF [15]) differs from the spatial routing solutions only in the heuristic function (e.g. areal distance or landmarks [13]). Correspondingly, we improve heuristic functions for uncertain mobility tasks by exploiting the spatio-temporal nature of the problems. We focus on applications that have SSRs as fundamental component of their domain. When scaling the settings, the state space grows exponentially with the number of SSRs. Soon, approximate solutions become necessary. A major topic is the state representation as input for deep neural networks. The position of the agent is a discrete node in the graph. When learning an embedding for the nodes (i.e., one-hot encoding) advantages to the table-based reinforcement learning are relativized. Related work discretizes the street network into grid cells [1, 26, 27, 28, 50]. Then, actions are to go north, south, east or west. The agent can hence decide only on rough areas, but it does not make routing decisions anymore. Our goal is to let the agent decide on the lowest possible level, i.e., which action to take next.

2.3 Applications

2.3.1 Parking Search

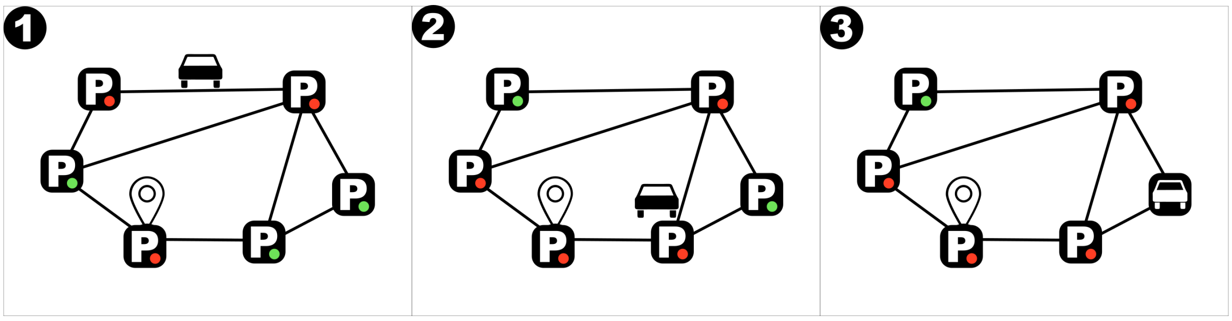


Figure 2.2: Illustration of a parking search. The P signs indicate a node with a parking spot. A green/red dot denotes that it is available/occupied. The target sign marks the target location of the driver. The car icon shows the position of the driver. (1) The driver heads the available parking spot to the right of the target location. (2) By the time that he/she arrives, the parking spot became occupied. Hence the driver attempts to move to the next best parking spot (3).

Finding an available parking spot close to the destination is a time-consuming task in many urban areas. The increasing individual traffic and the spatial limitations resulted in a shortage of parking spaces, especially in populated areas (e.g., restaurants, office district), as well as areas that host large events (sport, concert). With an uninformed search of parking spots, the driver searches slowly in a kind of random walk manner around the target. Parking search causes on average 30 % of urban traffic [48]. Accordingly, the driver slows down the traffic flow and produces unneeded CO_2 emissions. Moreover, the additional driving time and the slower traffic flow have harmful economic consequences. Some cities have installed on-street parking sensors to tackle this issue (e.g. Melbourne and San Francisco). Additionally, mobile applications and modern cars can detect parking activities [31]. Although applications with partial observation are even more challenging due to the partial and erroneous nature of the data, even if flawless information is available, the parking spots' states still change while driving according to a stochastic process.

Previous research in (partially observable) parking search tried to solve the problem with solutions based on the paradigm of time-dependent travelling salesman problem (TSP) [19, 20]. Solving the TSP is computationally expensive and only approximates the actual query, as it does not handle stochastic state transitions. We map the resource routing task to the framework of MDPs.

Other approaches suggest a reservation system for parking spots. In our opinion, such a system may be unfair and may end up being waste of resources. A reserved parking spot will be unoccupied yet it will not be available for other cars for a particular amount of time. Furthermore, a customers may have other business to attend and no longer need the spot, leaving an open spot unavailable for other people. It is likely that spots are reserved precautionary to have a spot just in case. On the other hand, someone might just need to drop something off while the spot is still empty, thus blocking the reserved space.

Banning individual traffic from cities would solve the issue of insufficient space for parking, however, it could result in an overload of the capacities of already overcrowded public transportation systems. Besides, it is usually unmanageable to avoid individual transport (e.g. delivery vehicles).

A parking spot is a resource having two states, available and occupied. The goal is to find an available resource as fast and as close to the target as possible. Hence, we refer to this task also as the *resource routing* task.

2.3.2 Travelling Officer Problem

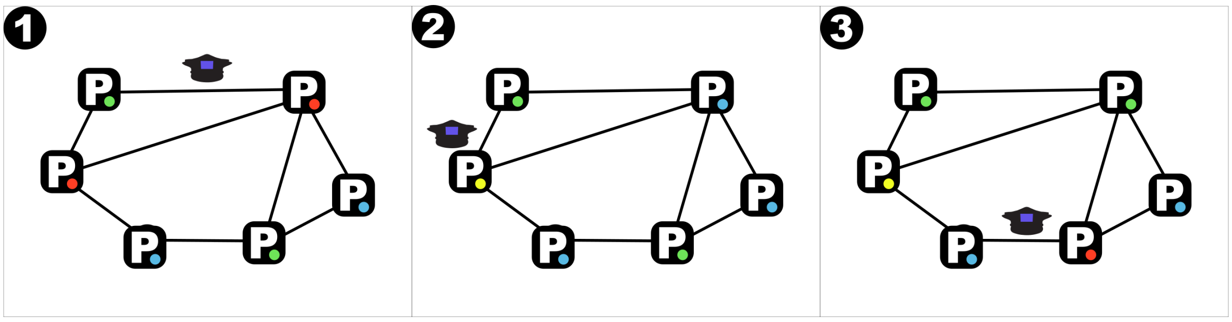


Figure 2.3: Blue, green, red, yellow dots indicate parking spots in free, occupied, violated and fined state, respectively. The officer fines the violation on the left hand side (1-2). When he/she arrives, the parking states changed and there is no violation left, so the officer moves towards the parking spot being in violation soon (3).

Due to the high demand and the low offer of parking spots in urban areas, many cities restrict parking spots with a maximum parking duration. This yields a fairer share of resources. Unfortunately, without controlling the restrictions, violations would frequently occur. Most large sites have parking officers patrolling. However, if the officer has to check every car with an uninformed search, human resources are used ineffectively. Various data sources, e.g. camera systems, mobile apps detecting parking events or movement data of vehicle fleets, could enhance the efficiency and effectiveness of the officer's observation. Some cities, e.g. Melbourne, San Francisco and Canberra (Manuka), have on-street parking sensors. The officer may exploit this information to detect overstays. Due to legal constraints, a parking officer may have to deliver the parking tickets by hand. Hence, the goal of this task is to maximize the number of issued parking tickets. Withal, the human resources are limited and it is paramount that a particular officer can cover a larger area.

Suppose the actual departure and arrival times of vehicles are known in advance. Then, the travelling officer problem would be an instance of a time-dependent travelling salesman problem. In reality, due to on-street sensors, we know the current state of the parking spots, but we do not know the future parking activities. Equivalent to the parking tasks,

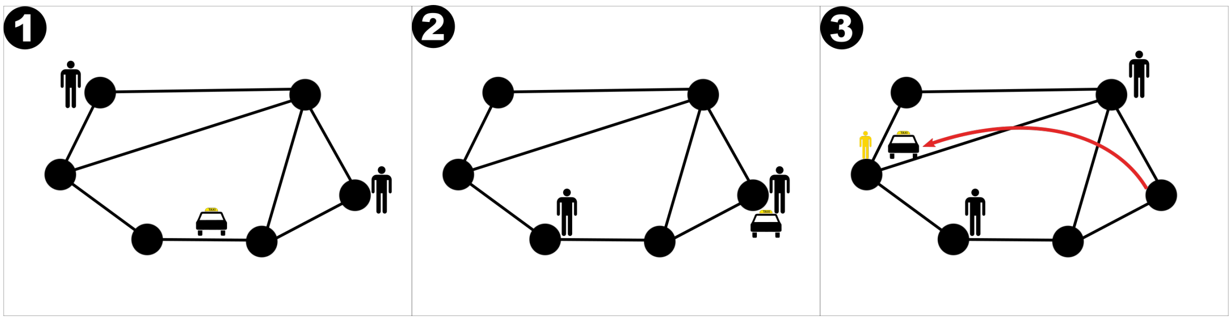


Figure 2.4: Illustration of a taxicab example. The taxi-icon shows the location of the agent. Human icons are passengers looking for a taxi. (1) The taxi chooses to pick up the passenger on the right-hand side. When he/she arrives (2) the other passenger is not available anymore, but a new passenger appeared. (3) The taxi driver brings his/her passenger to the destination location.

we assume a stochastic process behind the vehicle movements. Thus, we do not know if the car in violation is still available by the time the officer arrives. It becomes even more uncertain when we consider the parking spot states after more time has passed. The set of all possible follow-states after fining a car is vast. Therefore, we argue it does not make sense to plan too far into the future, i.e., solving a travelling salesman problem, as done in previous work [47]. Preferably, our policy should choose the parking spot, having the optimal expected value of accumulated future parking tickets. Again, this is a query for an MDP.

2.3.3 Passenger Dispatching

The personal transportation of passengers also plays an essential role in large cities. Regardless, if a single taxicab wants to minimize its idle time or a whole fleet seeks optimization, the goal is always to maximize the customers handled. In some settings it is even allowed to share a ride with other customers, e.g., [1, 26]. Existing variations of this task are determined by the availability and nature of the provided information. The demands of a taxicab with no information about potential passengers and other taxicabs, e.g., [5, 22], have different challenges than matching taxis and passengers with full knowledge about passengers, e.g. [50]. Nevertheless, in all manifestations an idle vehicle should be positioned as close as possible to a potentially new passenger, and minimize the waiting and driving time of passengers. Since a stochastic process behind the appearance of passengers is reasonable, it makes sense to solve this as an MDP.

2.3.4 Courier Service

The delivery of goods from one place to another is a service with increasing demand, especially in urban areas. An employee of the courier company riding a truck, car or bicycle receives incoming source and destination locations or customer requests. The type

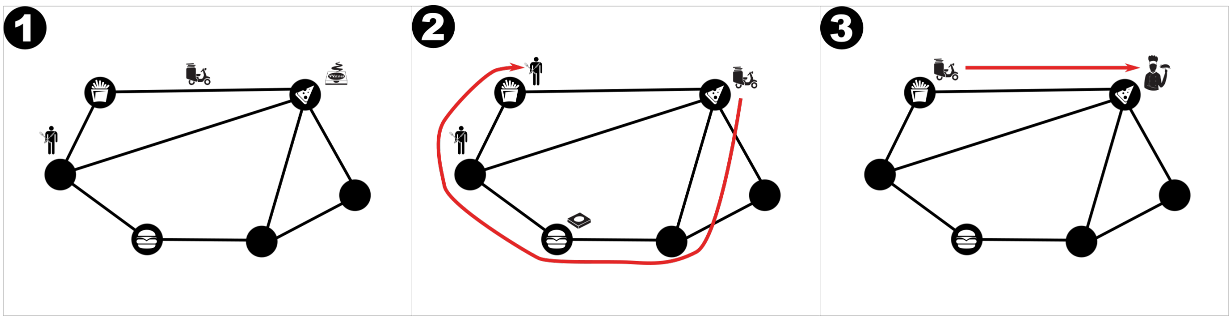


Figure 2.5: Illustration of a food delivery task, an example of a courier service. An employee of the courier company decides to pick up and deliver a pizza (1). When he/she enters the pizza restaurant, a new request from a burger restaurant turns up (2). The courier decides to pick up the burger on the way before delivering the pizza. Afterwards, he/she delivers the burger to the corresponding customer. Finally, the employee returns to the pizza restaurant because (3) he/she received the information that a pizza is about to be ready soon.

of goods is variable. Some couriers specialize in mail delivery, while others deliver food from restaurants to customers. Nevertheless, all courier companies share the property, that new customer requests arrive over time in a non-deterministic manner. Furthermore, a courier may decide to pick up multiple commodities before delivering them. At the same time, a courier aims to minimize delivery times.

The courier task is different from existing vehicle routing problems (VRP) [36, 49], where the target is to visit a predefined set of target locations. An instantiation of the VRP is the travelling salesman problem (TSP). But there are other variants, such as VRP with (soft) time windows, where the vehicle has to (or should in case of soft time windows) visit the targets at predefined periods. Some research assumes that new requests occur dynamically over time. In the courier task, however, we have pickup and drop-off locations. Delivery requests are usually not announced before the pickup time. VRPs typically try to minimize the costs, while in the courier setting, the agent aims to maximize the number of delivery requests satisfied in time. Furthermore, the courier can decide if he/she wants to deliver a parcel. In many VRPs the vehicle has to fulfill all requests, however, variants of the problem in which some deliveries cannot be fulfilled (with associated penalty, subject to minimization) have also been considered, e.g., [51].

2.3.5 Pursuit

In some scenarios, an agent wants to catch one or multiple moving targets. For instance, a policeman needs to arrest a criminal. Another example is finding a dog equipped with a GPS transmitter, after it had ran away. Sometimes the exact location of the target is known (e.g. by GPS, video cameras or a helicopter). However, one cannot know where it is going to move next. Hence, the target is modifying its location according to a stochastic process.

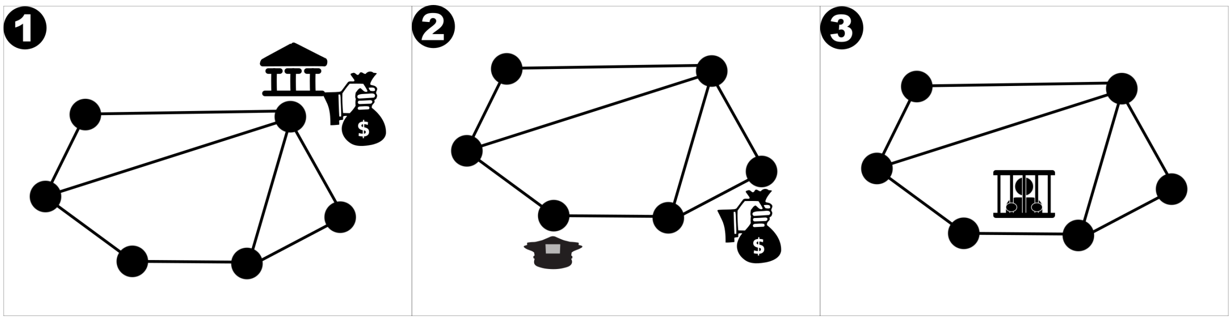


Figure 2.6: Illustration of a Pursuit. A thief robs a bank (1). Afterwards he/she takes flight and the police is alerted (2). The cops follow and catches the thief and imprisons him/her (3).

2.3.6 Summary

In the above examples, the objectives and potential user groups are very diverse. Nevertheless, they share common properties. First, each application operates on a user (agent) positioned on a street network. The agent wants to visit one or multiple locations, that we call resources, e.g. parking spot, pickup/drop-off locations or a moving ice cream cart. Some of those resources move over time or change their state according to a potentially unknown stochastic process. In this thesis, we present solution approaches for the applications mentioned above.

Chapter 3

Solving Mobility Tasks with Stochastic Spatial Resources

This chapter depicts the relations and the contributions of the papers that piece together this thesis. First, we model a task (resource routing) with real-world application (parking search) as an MDP and solve it with dynamic programming [44]. The experiments show that our new approach outperforms previous solutions by exploiting real-time information. The results indicate that the MDP-based solution can effectively utilize the real-time information to provide improved solutions. Then, we improve the dynamic programming solver [43] by focused Bellman updates, improved heuristics and pruning transitions with marginal probability. The new approach notably outperforms standard solver. Despite these significant improvements, the state space complexity remains exponential. Hence, we present approximate solutions based on replanning and hindsight planning [41]. The conducted experiments demonstrate that our approximated solutions are close to the optimum in environments in which the MDP is solvable within an adequate time period. Moreover, our replanning and hindsight planning approaches scale well with the number of SSRs, thus, enabling the quick computation of environments with SSRs in the margin of hundreds. In addition, we apply model-free deep reinforcement learning on the travelling officer problem [46]. The distance-based function approximation shares parameters along all SSRs. Consequently, it is sample efficient and transferable. In large and complex environments, we significantly outperform previous approaches and generalize it for all mobility tasks having SSRs [42]. Comparison with standard reinforcement learning baselines show that our solution is more sample efficient on all four example tasks. Finally, we present a solution and training environment for multi-agent and partially observable settings (taxi passenger dispatching) [5, 9]. Table 3.1 gives a concise summary of the main results and indicates the respective publication in which the findings were presented.

Research Question	Brief Description	Paper
How can we solve mobility tasks with full information about spatio-temporal elements in general?	The framework of Markov decision processes and reinforcement learning fits well to those environments. We envision that reinforcement learning and other Markov decision process solver will appear more frequently in future research when considering real-time information about spatio-temporal elements that affect the objective function.	[45]
How can we optimally solve the research routing task under full observation?	We model the research routing task with real-time information as Markov decision process and solve it with value iteration [4].	[44]
How can we efficiently solve the resource routing task if pre-computation is not possible?	We use the anytime approach with heuristic search Bouned Real-time Dynamic Programming (BRTDP) and improve the bounds. Furthermore, we skip very unlikely transitions to speed up updates.	[43]
Is it possible to approximately solve the Resource Routing task with hundreds of resources in real-time?	The Resource Routing task is not probabilistic interesting. Hence, replanning [29] and hindsight planning [53] performs well. We exploit the spatial nature of the task to make it scalable.	[41]
How can we compute a good policy for the travelling officer problem?	Previous research solved the problem with a time-varying travelling salesman problem. We model the task as a semi-Markov decision process and solve it with deep reinforcement learning.	[46]
Can we solve all spatial mobility tasks having SSRs with the same algorithm?	We define the new problem domain called Non-deterministic Spatial Mobility Tasks (NSMT) and integrate it to the framework of MDPs. We present a sample efficient reinforcement learning approach that is able to solve all problems of this domain without modification.	[42]
What is a good policy for searching new taxicab passengers when the taxis do not know the location of passengers and other taxis (GIS Cup 2019)?	We propose round-trips for idle taxi drivers. Choosing the next round-trip is done with a stochastic policy learned with a multi-armed bandit approach.	[5]
How can we train multi-agent environments with SSRs?	We provide a training environment for multi-agent resource search tasks with different levels of observation.	[9]

Table 3.1: Overview of the research questions and the corresponding summary of the work within each paper that is part of this work.

3.1 Resource Routing

Resource routing is the first spatial mobility task with SSRs that we address. In the problem set-up, an agent navigates on a street graph. Furthermore, there is a set of SSRs usually with two states, i.e., available and occupied. The states flip according to a stochastic process over time. The objective is to find an available resource as quickly and efficiently as possible. Depending on the application, acquiring an SSR may imply terminal costs. One example application is the parking search. As explained in Section 2.3.1, an agent is looking for an available parking spot. After the car is parked, the driver walks to the actual destination (terminal costs). Other applications are finding charging stations or free rental bikes.

3.1.1 Defining an MDP for Resource Routing

The states of resources in the resource routing task alter over time according to a stochastic process. Hence, the optimal solution must consider all possible future states, generated by the stochastic process, which, as described below, is what an MDP does. Hence, we argue that the optimal solution can be computed by solving the corresponding MDP. Modeling an instance of the resource routing domain as an MDP requires expert knowledge about the stochastic processes (model assumptions). In [44], we provide an MDP model for the resource routing task under full observation (i.e., real-time information about the SSRs' states). We propose to represent a state as a combination of the driver's current node and the availability of each SSR, which naturally implies that the state space expands exponentially with the growing volume of SSRs. The set of actions are the edges in the graph, yet for a given state, only adjacent edges are allowed. An MDP additionally defines a transition function, i.e., the probabilities for each possible follow state given the current state and action. For this purpose, we take two simplifications. First, we assume independence between SSRs. Second, we propose to model the dwelling time from occupied to available and vice versa exponentially distributed. Since the exponential distribution is memory-less, the probability that a parking spot is available at a given point in time is a continuous-time Markov chain. This is in accordance with [19].

In resource routing, the duration of different actions vary. Meaning, the distances and maximum speed between two nodes in the graph are variable. Hence, the driver needs a variable travel time for each edge. On the other hand, an MDP assumes a discrete and uniform duration for every step. If this is not the case, we have a semi-Markov decision process (SMDP). The discrete-time SMDP framework [17] allows to model the time between two decisions as a random variable. The two Bellman update rules differentiate only on how to discount future rewards (or costs). In the resource routing setting, we do not need discounting, because a second in the future counts as much as the current second. Accordingly, we set the discount factor to 1, implying that the MDP is mathematically equivalent to the SMDP.

In [44], we solve the MDP with value iteration [4], a well known dynamic programming approach for solving MDPs. Nevertheless, only a few parking spots are manageable due to the exponential growth of the state space. Therefore, we propose to pool all parking spots on one edge and pre-compute policies for small areas which are within walking distance of the actual target.

3.1.2 Bounded Real-time Dynamic Programming

Sometimes, pre-computation of policies for multiple areas is impractical. For instance, suppose there exists a target node, where the driver eventually wants to go after acquiring a SSR, and the objective is to minimize the overall time until the driver reaches the target node. Then, we would need to solve the MDP for every target node in advance, which is impractical. Thus, a solver that can retrieve a good policy in real-time is preferred. In [43], we propose to define the parking search problem as a stochastic shortest path – a subset of the MDP class. Thereupon, we can apply real-time dynamic programming (RTDP) [2] approaches, which update only states that likely appear for a given start state. More precisely, we use bounded RTDP (BRTDP) [32]. Hence, the updates are much more focused. We can directly apply BRTDP on our resource routing MDP model. It already helps as it reduces the runtime until convergence. However, especially the standard bounds do not scale with the number of resources. Thus, we eliminate the exponential factor of the runtime of both the lower and the upper bounds for the optimal expected costs of a state. Nevertheless, since every state has an exponentially increasing quantity of subsequent states (w.r.t. SSRs), handling even a single update becomes rapidly infeasible. Since the probability for a succeeding state is the product of probabilities for the respective resource transitions of the continuous time Markov chains, there is a vast amount of subsequent states with numerically irrelevant probability.

To tackle this issue, we introduce a novel algorithm, scanning for the minimal set of states such that the cumulative probability of the remaining events is smaller than a threshold. A naive approach would sort the set of transitions by the probability value and return the states with the highest probability until the sum of probabilities is large enough. The difficulty is that we do not want to iterate over every follow-state. Otherwise, we do not have any benefit. Consequently, we need an algorithm that returns the highest probable transitions without explicitly listing and sorting them. We propose a best-first search that guarantees to return follow-states with decreasing transition probability. By assuming independence between SSRs, we know that the transition with the most likely probability value is the one where each SSR has its most likely Markov chain transition. This follow-state is the root node of our search graph. Subsequently, we expand the root by adding a child for every possible separate SSR state switch. We add these follow-states to a priority queue and prove that the transition with the second-highest probability is in the priority queue. Then, we do the expansion for this node, get the follow-state with the highest value, and so on. Since most states have a numerical marginal, we can retrieve the highest likely transitions without processing most of the follow-states. Adding this algorithm to the real-time solver improves the runtime notably and only slightly decreases its quality [43].

3.1.3 Modeling Agent Locations

In contrast to [44], we define the location of the agent in [43] as an edge, not a node. An edge contains more information, i.e., the source direction. As a result, we can penalize u-turns, as they take more time than other turns. Furthermore, resources (e.g. parking spots) are typically located on edges, not nodes. However, there are about two to three times more edges, resulting in a larger state space. Overall, both attempts are valid and have their advantages and disadvantages. Formally, both fit well to the MDP framework.

3.1.4 Probabilistic Interesting

In the previous section, we proposed solutions that efficiently return (epsilon-)optimal policies for the resource routing task with dynamic programming. Since the MDP state space grows exponentially, no matter how efficiently we solve this problem, the runtime grows exponentially. Hence, it does not scale beyond around ten SSRs. However, there are applications for which this is not sufficient. For instance, a city can have hundreds of relevant parking spots. Therefore, we seek reliable and scalable approximations. We know that not every MDP is "probabilistic interesting" [29]. Intuitively, an MDP is "probabilistic interesting" if it contains states similar to the following example:

Suppose we are in a state with two actions, o and p. Action o has two outcomes. The first and most likely leads to the goal with high rewards. The second, still quite probable, outcome leads to a state from where it is impossible to reach the goal. Action p always results in a goal state with medium reward.

If the MDP is not probabilistic interesting, we can use the approximation replanning [29], i.e., we relax the probabilistic problem such that it becomes deterministic. A common approach is to assume that the most likely outcome will happen. The deterministic problem has a much lower computational complexity. Hence, the agent acts according to the deterministic plan, and when an unexpected state occurs, we replan.

What happens if we apply replanning on the probabilistic interesting example from above? The agent would choose action o. But this choice is poor as we will not reach the goal with moderate probability. Nevertheless, we argue that resource routing is not probabilistic interesting. Typically, the driver can (almost) undo this action by driving back. And, there exists no state from which you can never acquire an SSR. For this reason, we propose a replanning and a hindsight planning approach to cater to scalability in the parking search scenario [41].

Hindsight planning [53] revises some shortcomings of replanning, but it has a higher computational complexity. The basic idea is to sample many possible deterministic futures, solve them, and estimate the expected future costs. As this assumes knowledge about future development while solving the determinization, it is an optimistic approximation of the real expected return. Then, the agent greedily chooses the best action.

By choosing replanning and hindsight planning as the solver, we relaxed computational complexity w.r.t. the number of states, but the exponential state complexity still remains. Consequently, we present equivalent approaches that do not depend on the number of SSRs.

This is possible because we assume one specific development of the SSRs. Accordingly, the SSRs do not need to be part of the state space. Further, we can exploit some spatio-temporal properties of the resource routing task, and have the state-space become linear in the number of graph nodes.

If there is currently no available SSR, replanning does not return any solution, because the most likely transition after every action is that no SSR changes. Then, the planner has no chance to find the goal state. Furthermore, it is a valid policy to drive to a good SSR and wait until it is available. Accordingly, we alter the cost model of the deterministic planning task, such that an occupied SSR has an action allowing us to wait for the SSR to be available. The costs for this action are the expected waiting time until the agent can acquire this SSR.

Since in large settings most of the time after every step at least one parking spot changes its state, in replanning, recomputation of the deterministic solution would be persistently necessary. We use the D*-Lite algorithm [23], which updates the values only if the change is relevant. This way, handling hundreds of parking spots becomes feasible. We show that our replanning approach is quite close to the optimal solution in settings where the MDP is still solvable.

One weakness of replanning is that it independently considers parking spots. Hence, it will prefer one slightly better parking spot to two spatially close parking spots. As shown in [41] for hindsight planning, this is not an obstacle. Due to the sampling of deterministic futures, hindsight planning will notice that most of the times the direction towards the two parking spots is optimal. In an inhomogeneous synthetic setting, hindsight planning performs slightly better than replanning. However, in more realistic environments, both approaches perform about the same, yet hindsight planning consumes more computational capacities.

To conclude, we propose to use replanning and hindsight planning to solve the resource routing task. We adapt the general-purpose approaches to exploit expert knowledge of this task. In settings where BRTDP still converges in a reasonable time, our improved replanning and hindsight planning approaches are close to the optimal solution. Furthermore, when applied to hundreds of parking spots, the approaches generate good policies while completing within a reasonable amount of time.

3.2 Resource Collection

In this section, we discuss the methods proposed for the *resource collection* task, which is quite similar in spirit, and can be formalized in a related manner to the resource routing task. Both have an agent seeking for SSRs while navigating on a street network. However, in resource collection, there is no terminal state transition after consuming an SSR. Instead, the agent is positively rewarded for collecting an SSR. Following, the SSR is non-collectable for an unknown period. The objective of the resource collection task is to collect as many resources as possible in a pre-defined period. In summary, the resource collection task is a finite horizon MDP consisting of a street graph, an agent navigating on it and a set of

SSRs. An example application is the travelling officer problem (cf. Section 2.3.2).

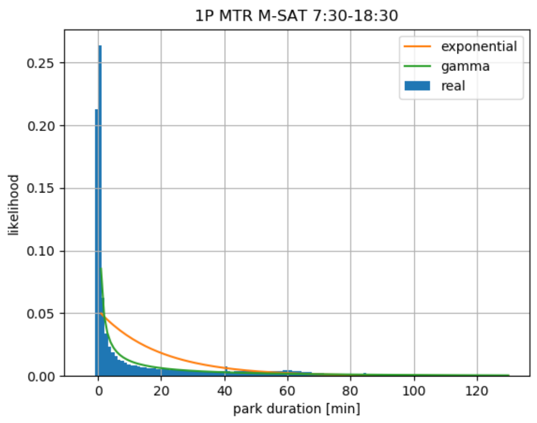
The objective of resource collection is to maximize the number of SSRs collected. Hence, it is not indicative using the MDP with the cost terminology. It is possible to define an MDP such that the target is to maximize the expected future rewards or to minimize the expected future costs. So far, we used the cost terminology, since it is common in the spatial community to minimize the travel. In the resource collection task, we want to maximize the collected resources. Hence, we award every collection event positively.

As mentioned earlier, the problem definitions of resource routing and resource collection are related. The application of the resource routing solution to the resource collection task sounds therefore reasonable. However: (1) standard dynamic programming approaches (as in [44]) do not scale due to the exponential growing state space. (2) Unlike the resource routing task, resource collection is no stochastic shortest path. Resource collection is a finite horizon MDP with no termination except the end of the working day. Accordingly, the trials are very long and diverse. RTDP approaches, as proposed in [43] for resource routing, would need too much time to converge in this setting. (3) Creating a deterministic future of the problem (as done in [41]) does not substantially reduce the complexity. If the future is known, the resource collection task is a Vehicle Routing Problem (VRP) with time windows – an NP-hard problem [25]. It is possible to use approximations for solving VRPs [21, 24, 34] and VRP with time windows [18]. However, it is more promising to use reinforcement learning with function approximations for solving the actual problem rather than using the same approach to solve a relaxed version of the problem. Furthermore, we argue it is questionable to plan too far in the future (complete working day) in a deterministic manner if the set of all possible futures is vast, and every potential eventuality would result in separate plans. Consequently, we propose to use model-free reinforcement learning with function approximation to solve the resource collection task.

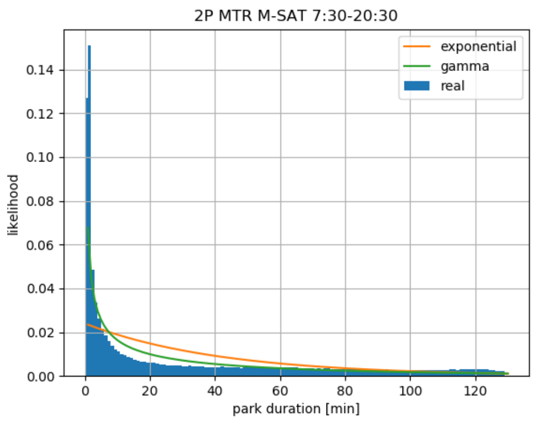
3.2.1 Model-free Deep Reinforcement Learning

All approaches mentioned hitherto, require a model assumption. It is irrelevant whether we demand the model to solve the MDP directly or to choose/sample deterministic futures. Either way, the assumptions likely deviate from reality. In some cases, a historical dataset of the development of stochastic spatial resources exists. For instance, the city of Melbourne published the parking events from January 2011 until May 2020 (as of September 2020). Therefore, we can check how good our model assumptions for the parking setting are. To recall, we assumed that the parking times are exponentially distributed. The benefit of this assumption is that the distribution is memory-less. In Figure 3.1 we compare the exponential distribution with the Melbourne parking dataset of 2019[1]. At first glance, the real data (blue) looks exponentially distributed. However, one can see that the likelihood decreases much faster than in the exponential distribution (orange). Figure 3.1 shows that another distribution, namely the gamma distribution (green), fits much better. We fitted the parameters of this distribution, such that the variance and expectation is equal

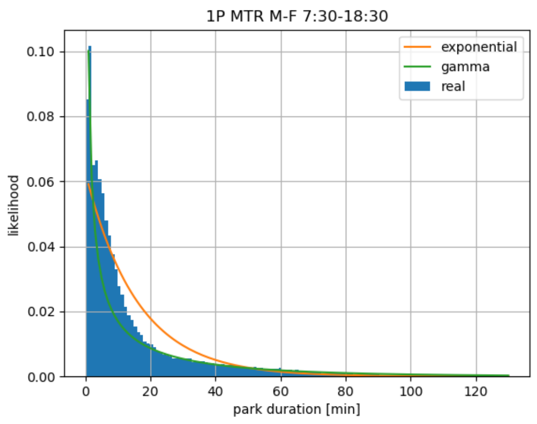
[1]`https://data.melbourne.vic.gov.au/Transport/On-street-Car-Parking-Sensor-Data-2019/7pgd-bdf2`



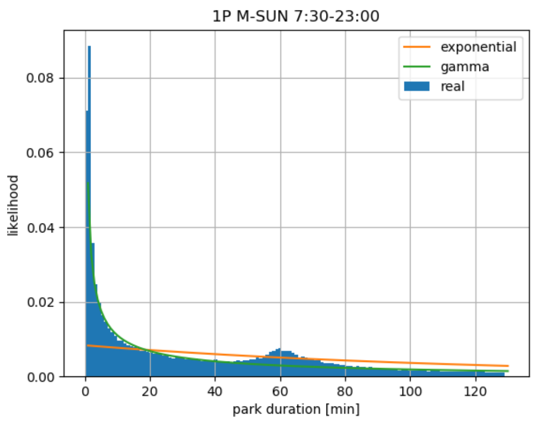
(a) 1 hour (Monday to Saturday 7:30 - 18:30)



(b) 2 hours (Monday to Saturday 7:30 - 20:30)



(c) 1 hour (Monday to Friday 7:30 - 18:30)



(d) 1 hour (Monday to Sunday 7:30 - 23:00)

Figure 3.1: Histogram of vehicle parking times in Melbourne. The subplots (a), (b), (c) and (d) represent different park signs. An exponential distribution and a gamma distribution is fitted for the most frequently used parking signs. The exponential distribution was chosen so that the expected value corresponds to the actual distribution. For the gamma distribution, the expected value and the variance are fitted to the data set. The title of each plot represents the parking sign. The X-axis displays the time in minutes and the y-axis is the likelihood.

to the real data. However, the gamma distribution is not memory-less. The increasing likelihood around the maximum parking duration further emphasizes that the real data is not memory-less. This implies that the MDP state requires the present parking duration of a vehicle on its associated parking spot to be Markovian. Hence, the state space would be continuous. This, in turn, excludes standard dynamic programming approaches from the set of possible solvers.

We note, however, that if, if the real-world dataset is representable, one can directly apply model-free learning approaches to this dataset. The prerequisite for this is the assumption that the stochastic SSR dynamics of the dataset will not be affected by the agent. In the travelling officer problem example, this means that we assume that an assigned parking ticket does not shorten or extend the parking duration of the car. We argue that this is true for most of the cases because the driver will notice the ticket only when he/she departs. Once we made this assumption, we can replay the historical data and let the reinforcement learning agent train on top of this simulation. Formally, we choose a random day within the dataset and jump to its starting hour. We look up the current state of each SSR at the starting time. The agent has an arbitrary node on the graph. Whenever the agent moves, we add the travel time to the simulation time and look up the new state of the SSRs. When the agent collects an SSR, it changes the SSRs collected property to true. We continue similarly until the end of the working day. By this procedure, there is no need to create stochastic processes adapted to the real data. Hence we introduce less bias. The model-free reinforcement learner implicitly learns the stochastic processes of the dataset.

Based on this reasoning, we argue that model-free reinforcement learning is an excellent choice for solving the resource collection task. In [46], we propose a Deep-Q-Network-based (DQN) [33] solution, i.e., a famous deep reinforcement learning approach, with a specialized function approximation for the resource collection task. Since we have non-uniform action directions and a discount factor < 1, we need to regard the semi-Markov characteristics. We propose higher-level temporal abstractions, i.e., moving directly to a specific resource. The agent's location (a node) is a discrete entity. Hence, a naive approach would be to learn an embedding for each node. We decided against it, because that way the function approximation is unable to generalize experience between two different locations, even if the nodes are next to each other. Our proposed function is distance-based. Hence, the artificial neural network can generalize experience to states where the distances are similar. Another benefit of this is the fact that it is independent of the graph, allowing the applications of the trained weights without further training to previously unseen street graphs. We compare our solution with different deep reinforcement learning baselines and previous approaches for the travelling officer problem. The solutions proposed in the literature [47] are time-varying travelling salesman problem solvers (replanning) and a heuristic greedy agent. If the number of simultaneous violations is small (e.g. in tiny areas), the approaches from literature show good results. However, when we increase the area size, solving the travelling salesman problem becomes prohibitively costly, and the greedy heuristic becomes too weak. Consequently, our approach significantly outperforms the exisiting baselines in large areas. Furthermore, we show that training on a small graph and applying it to large environments is possible without further training.

3.3 Non-deterministic Spatial Mobility Tasks

In [42] we introduce and formally define a new problem, which combines all the applications mentioned in Section 2.3 into one problem definition, which we call non-deterministic spatial mobility tasks (NSMT). The applications have very different dynamics and objectives. Nevertheless, they all share some common properties, i.e., they have SSRs and the intent is to move an agent around in a street graph. Although we can assume a subsequent node to be deterministic (i.e., driver executes only instructed turns), the SSRs are typically not. All SSRs have a location, and the agent might alter the properties if it is nearby. By doing so, the agent may (positively) affect the rewards received and sometimes modify the agent's properties. While some NSMTs terminate after "consuming" a resource, others have a fixed horizon. Furthermore, besides the location, an agent may have further properties that may or may not be non-deterministic, e.g. passengers on board or charging level. To conclude, all tasks consist of:

- A street graph
- An agent with a location (node or edge) and other potentially non-deterministic properties altered by a stochastic process
- SSRs have a position and features, both of which are potentially non-deterministic

Then, the agent can choose an action (the next edge) and alter SSRs. The objective is to maximize the (stochastic) rewards (or minimize the costs) received. The only differences are the shape of the properties and the stochastic processes which alter the non-determinisms.

Notice that the objective function for the resource routing task in [42] looks quite different from the objective function in [43, 44]. In the more general framework, we prefer to use reward (higher is better). One method to transfer costs to rewards is to multiply by -1 and to maximize negative rewards. However, this reward function does not fit to our function approximation well. In preliminary experiments, the agent tends to learn to choose shorter distances before recognizing that it has to move the long way to achieve a goal. A sparse reward signal when reaching the goal performed much better as it seems easier to learn that an available SSR gives a positive reward. Yet, if we want to minimize the travel time, we have to use a discount factor with a value smaller than one. Then, time is decayed exponentially instead of linearly. Since both decays are monotone, the optimal policy is the same.

3.3.1 ORIENTATION: One Solver for all Applications

So far, we had to develop a specialized solution for every task. Adapting the solver to every NSMT needs a lot of expert knowledge. For a broader applicability, we present a general solver approach for all NSMTs. Reinforcement learning has the capability for this. Its goal is to find a solution for every sequential decision making task under uncertainty. However, table-based approaches and general-purpose function approximations do not scale when applied to NSMTs. Hence, we use a distance-based function approximation optimized for navigation on graphs with SSRs [42]. Further, we propose to make use of the options framework, i.e., add higher-level temporal abstractions (directly routing to a specific node in the graph). Higher level temporal abstractions, i.e., hierarchical reinforcement learning, are the key to solving large MDPs. With our proposed set of options, the agent does not need to learn how to route from one location to another, but it can choose the option for doing so. This speeds up the exploration process and is therefore more sample efficient.

We call our new framework ORIENTATION. It is transferable to previously unseen networks in the same domain. The transferred network performs well as long as the stochastic processes of the SSRs are similar. If they differ too much, the implicitly learned assumptions become invalid, and further training is required. Nevertheless, despite general-purpose reinforcement learning approaches, ORIENTATION's function approximation has a constant number of parameters regardless of the network size and number of SSRs, allowing for great flexibility. To conclude, ORIENTATION is a model-free RL approach applicable to all single-agent NSMTs, with the property of being sample efficient and transferable.

3.4 Multi-agent and Partial Observations

All proposed approaches so far were on a single agent and assumed full observability. However, some NSMTs are indeed multi-agent settings, e.g. the taxicab task. In our submission to the GIS Cup 2019 [5], we apply reinforcement learning on the taxicab task. In the setting defined by the competition, the goal was to optimize thousands of taxis such that the idle driving times are minimal and the passengers dispatched are maximal. However, the taxis are unaware of the positions of the other vehicles. A centralized agency automatically assigns passengers to the closest taxicabs within an acceptable radius. We trained a model-free one-step reinforcement learning approach (multi-armed bandit) with temporal abstracted actions and the current time as observation. The weights were pre-trained with a statistical model. For further research in this area, we designed a framework [9] to train multiple agents in NSMTs (in particular Taxicabs), where one can decide between different levels of observability.

Chapter 4

Concluding Remarks

In the following, we summarize the contributions and offer directions for future research.

4.1 Conclusion

Due to the increasing amount of sensors deployed in smart cities, more information becomes available about the current state of stochastic resources. One example is the parking search. Ignoring parking sensor values would cause extra driving to search for a parking spot and increase the emission of CO_2 since vehicles take more time on the street for an uninformed search of parking spots in crowded urban areas. Other origins of knowledge are likewise crucial for future transportation tasks. For instance, mobile applications collect data of food or passenger pickup and drop-off locations, parking activities or traffic congestion. We argue that the future will bring more and more spatial applications where an agent needs to react according to stochastic spatial resources (SSRs). With this work we show how to model the resource routing task as an MDP and solve it (epsilon) optimal. The dynamic programming based solvers are well-suited in case pre-computation is possible, the number of resources to consider is manageable, and we want to have quality guarantees. For settings where pre-computation is not possible, and the number of resources are smaller or equal to ten, we present a more efficient anytime approach. Often, a guaranteed optimal policy is not necessary, but we want to consider more resources. We show that resource routing is not probabilistic interesting and propose solutions based on replanning and hindsight planning. Both approaches are close to the optimal solutions where the MDP is still (epsilon) optimal solvable. However, replanning and hindsight planning have a reduced runtime complexity, which allows resources in the scope of hundreds without a problem.

In other spatial mobility tasks with SSRs (e.g. resource collection), the determinization is not efficiently solvable, or it is not “probabilistic interesting”. Additionally, previous methods require expert knowledge about the specific task and stochastic processes. With Model-free reinforcement learning, we can overcome this issue. We present a reinforcement learning approach that solves the resource collection task and subsequently, generalize it to a single framework that can optimize the policies for all non-deterministic spatial mobility tasks (NSMT) without any alteration. Our novel neural network architecture allows transferability to previously unseen street graphs and resources.

4.2 Future Work

The approaches proposed so far mainly focused on single-agent settings with full observations. Some sensor data, however, might be incomplete. For instance, suppose a mobile app tracks parking activities. Not every driver has this app installed or activated. Hence, only a fraction of the information is available. Also, there may be taxi passengers that do not request a ride over a mobile application. Due to an incomplete state description, further development is dependent on the history of observations. Hence, the value function becomes more complex and needs to get the history of resource states and potentially agent positions as input. In the case of function approximations, one can do this with recurrent neural networks. But there might be a smarter - potentially problem-specific - ways to do this, which are more sample efficient. If the model is known, it is possible to consider approximating the partially observable MDP (POMDP). Typically, this problem has a continuous state space with an exponentially growing dimensionality, which is why replanning approaches might become more tricky.

Furthermore, in many multi-agent NSMTs, a vehicle fleet tries to optimize a common objective. For instance, a taxi company tries to maximize the number of passengers dispatched. When we optimize this objective, the taxis share the reward. Hence, an empty cab in the middle of nowhere receives positive gratification regularly. But the positive reward does not come from his weak policy. In literature, this is called the credit assignment problem. To overcome this issue, one can optimize the agents independently. In many applications, this will be close to the optimum [8]. However, in general, the "price of anarchy" can be larger, leading to the question, how to tackle the credit assignment problem.

In this thesis, we provide solutions for NSMTs, having resources that affect the future reward positively when the agent is at the right time nearby. Notwithstanding, there are stochastic elements in a road network affecting the agent negatively, e.g. traffic congestion. Due to the structure of the options and function approximation, the agent has difficulties to avoid specific nodes. Therefore, one might relax the restriction that options are the shortest paths to a given node. Preferably, it can be any path to the node. But this dramatically increases the number of potential opportunities. The approach needs to learn an understanding of sound paths.

Another direction is combining the planning and reinforcement learning approaches. After training and convergence, inference of a reinforcement learning approach is fast. Planning, on the other hand, does not need time for off-line training, but it takes much more time on-line. Yet, by knowing the current state, planning can focus on the currently relevant parts of the state space, whereas reinforcement learning has to perform well on all possible states. One can consider reinforcement learning as the instinct and planning as the thoughts [12]. Combining both approaches should hence be able to improve the overall performance. However, the question is how to design this interaction and how much planning is allowed/necessary. Ideally, one can integrate a real-time planning approach to our option based reinforcement learning framework.

Appendix A

Publications

A.1 Dynamic Resource Routing using Real-Time Information

Publication

Sebastian Schmoll and Matthias Schubert. Dynamic resource routing using real-time information. In Michael H. Böhlen, Reinhard Pichler, Norman May, Erhard Rahm, Shan-Hung Wu, and Katja Hose, editors, *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*, pages 501–504. OpenProceedings.org, 2018

DOI: `https://doi.org/10.5441/002/edbt.2018.57`

Contribution

After discovering the Melbourne and Manuka (Australia) parking duration dataset, Sebastian Schmoll proposed the idea of modelling the parking search task with real-time information as MDP. Sebastian Schmoll suggested and implemented the model and value iteration solver and run experiments. The source code extends the existing NAVIGAZELLE framework. Matthias Schubert contributed in discussions with thoughts. Both, Matthias Schubert and Sebastian Schmoll, composed the text of this paper.

A.2 Dynamic Resource Routing using Real-Time Dynamic Programming

Publication

Sebastian Schmoll and Matthias Schubert. Dynamic resource routing using real-time dynamic programming. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4822–4828. ijcai.org, 2018

DOI: `https://doi.org/10.24963/ijcai.2018/670`

Contribution

Sebastian Schmoll proposed to solve the parking search problem with real-time information with the Real-time Dynamic Programming (RTDP) algorithm. Sebastian Schmoll implemented several extensions of RTDP and found out that Bounded RTDP performs best. Sebastian Schmoll developed improved heuristics and the transition pruning strategy. He wrote down the lemma and proofs have the TopProbabilities algorithm after discussions with Matthias Schubert. Sebastian Schmoll extended the source code from the previous paper with the new approaches and run the experiments. Matthias Schubert contributed in discussions with thoughts. Both, Matthias Schubert and Sebastian Schmoll, composed the text of this paper.

Erratum

- In Algorithm 1, in the fourth line from below x shall be replaced with s'.
- In Algorithm 1, in the second line from below x shall be replaced with s_0.

A.3 Vision paper: reinforcement learning in smart spatio-temporal environments

Publication

Sebastian Schmoll and Matthias Schubert. Vision paper: reinforcement learning in smart spatio-temporal environments. In Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong, editors, *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, pages 81–84. ACM, 2018

DOI: `https://doi.org/10.1145/3274895.3274963`

Contribution

The vision that an increasing amount of real-time information about dynamic elements in spatial navigation tasks will be available in future, has been developed in discussions between Matthias Schubert and Sebastian Schmoll. Since future developments of such dynamic elements are uncertain, Sebastian Schmoll proposed to model them as Markov decision process, which can be seen as stochastic generalization of usually routing.

A.4 Scaling the Dynamic Resource Routing Problem

Publication

Sebastian Schmoll, Sabrina Friedl, and Matthias Schubert. Scaling the dynamic resource routing problem. In Walid G. Aref, Michela Bertolotto, Panagiotis Bouros, Christian S. Jensen, Ahmed Mahmood, Kjetil Nørvåg, Dimitris Sacharidis, and Mohamed Sarwat, editors, *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019*, pages 80–89. ACM, 2019

DOI: `https://doi.org/10.1145/3340964.3340983`

Contribution

Literature research brought Sebastian Schmoll to the property "probabilistic interesting". As a result, Sebastian Schmoll developed the idea that resource routing is an MDP, but is well approximated by replanning methods. Further research leaded Sebastian Schmoll to hindsight planning. Sabrina Friedl proposed to use the D*-Lite algorithm. Sebastian Schmoll have developed and implemented the solutions (based on NAVIGAZELLE) and executed the experiments. The effect of the Indecisive Hindsight Planner and its solution has been discovered and developed by Sebastian Schmoll. Matthias Schubert helped with hyper-parameter optimization. Sabrina Friedl, Matthias Schubert and Sebastian Schmoll contributed the text to the paper.

Erratum

In Algorithm 2, line 7 shall be replaced with:

$$P(r = available) = \begin{cases} T_{a,a}^{\text{timeBefore}}(r) & \textit{if } r \text{ is available} \\ T_{o,a}^{\text{timeBefore}}(r) & \textit{if } r \text{ is occupied} \end{cases}$$

A.5 Semi-Markov Reinforcement Learning for Stochastic Resource Collection

Publication

Sebastian Schmoll and Matthias Schubert. Semi-markov reinforcement learning for stochastic resource collection. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3349–3355. ijcai.org, 2020

DOI: `https://doi.org/10.24963/ijcai.2020/463`

Contribution

Sebastian Schmoll contributed to this paper the idea to model the travelling officer problem as a Markov decision process and to use temporal abstractions. Sebastian Schmoll proposed and developed the function approximation and feature representation. Further, the environment and experiments have been developed and executed by Sebastian Schmoll. Matthias Schubert helped with hyper-parameter optimization. Both, Matthias Schubert and Sebastian Schmoll, composed the text of this paper.

A.6 ORIENTATION: Option-based ReInforcemENT learning for spatial navigATION

Publication

Sebastian Schmoll, Sabrina Friedl, and Matthias Schubert. Orientation: Option-based reinforcementlearning for spatial navigation. Published in appendix, October 2020

Contribution

Sebastian Schmoll proposed to separate the task of routing from the actual spatio-temporal task. Therefore, he proposed to use the options framework with target options. Sebastian Schmoll generalized the approach from [46] and implemented the source code in a new framework. Experiments have been performed by Sebastian Schmoll. Sabrina Friedl, Matthias Schubert and Sebastian Schmoll composed the text of this paper.

Publication below:

ORIENTATION: Option-based ReInforcemENT learning for spatial navigATION

Sebastian Schmoll, Sabrina Friedl, Matthias Schubert
LMU Munich
Germany

October 13, 2020

Abstract

In many tasks involving spatial navigation in road networks, the environment contains stochastic resources making it non-deterministic. Examples for resources are parking spots or charging stations, passengers for taxicabs, as well as moving objects like other vehicles, etc. Even if a smart city infrastructure provides the current status of these resources, future developments are uncertain and often unknown. Finding optimal navigation policies in these settings is often not possible due to the exponential growth of the state space w.r.t. the number of resources. Hence, reinforcement learning methods using function approximations are essential to scale solutions into realistic settings. In this paper, we present a framework for learning efficient policies with reinforcement learning, which applies to any task involving spatial navigation in environments as described above. More precisely, we propose to use the options framework and introduce a set of options that make use of established route planning methods such that learning can focus on higher-level goals. Furthermore, we present a sample efficient function approximation that applies to a wide range of mobility tasks and is transferable to previously unseen graphs. In our experiments, we show that one can apply the learned function approximation without further training to previously unseen road networks in the same problem set.

1 Introduction

Recently, reinforcement learning (RL) has started receiving increasing attention in the spatio-temporal domain [25]. Applications like the taxi dispatching [10, 12, 30], ride-sharing [1, 11], and travelling salesman problems [8, 9] took advantage of recent developments in general RL. Besides, the growing availability of location-based data in smart cities further stimulates this trend. Taking into account more data can potentially improve the execution of spatial tasks, e.g. by minimising search times, lowering traffic or reducing pollution.

All of the mentioned tasks have in common that an agent navigates a road network and visits locations to receive a reward. Let us note that the availability of the reward might change over time, e.g. a new passenger disappears. Due to the varying availability, the state space usually grows exponentially concerning the reward locations. To reduce this complexity, a variety of approaches works on a grid overlay of the map, e.g. [11, 30]. This way locations and resources are summarised into grid cells and actions are limited to visiting neighbouring grid cells. However, movement along with grid cells often yields only a rough approximation of navigating a road network as travel times between various locations in two adjacent cells may strongly vary.

On the other hand, standard RL approaches usually suffer from the large action space (edges in the graph) when trying to learn a policy for directly navigating a road network. General-purpose function approximations cannot share experience across taken edges, resulting in the need for a large number of training steps.

In this paper, we define *non-deterministic spatial mobility tasks (NSMTs)* and present a novel general framework for solving NSMTs where an agent directly navigates through a road network by choosing road segments as actions. In particular, NSMTs include all applications in which an agent traverses the network and affects the reward positively by being at the right location at the right point of time. Examples for these tasks include searching for free parking spots, pick up and deliver goods as well as chasing other vehicles. In all three cases, the environment contains what we call *stochastic spatial resources (SSR)* which require non-deterministic planning techniques. The non-determinism arises from the stochastic behaviour of SSRs: parking spots might change their availability over time, new delivery requests may occur and other vehicles constantly change their position in the network. Abstracting from these examples, all SSRs have in common that they provide a reward at certain locations for certain points in time. Furthermore, the behaviour of SSRs is assumed to be controlled by an unknown statistical process. Let us note that this is different from related vehicle routing problems [28] which rather work with static spatial resources having known locations and known time-intervals. In contrast, in our setting, the agent has to learn the behaviour from experience and even an optimal policy might miss rewards due to not reaching an SSR in time. When modelling spatial environments with SSRs, we can observe that even SSRs with a fixed position and the simple attribute of being available or not result in exponential growth of the state space w.r.t. the number of SSRs.

To solve these tasks efficiently, we present a function approximation whose number of parameters does not increase with the number of SSRs. Instead, the function approximation shares parameters when computing the impact of each SSR. Hence, a major benefit of our method lies in its sample efficiency. Another important characteristic of our approach is that it models tasks as semi-Markov decision processes (SMDPs). By doing so, we can allow actions, like traversing a road segment, to have varying durations. To avoid that an agent has to learn the subtask of routing as well, we propose to use temporal abstractions and define options that route to any specific node in the graph directly on the shortest path. To this end, we make use of deterministic routing methods like [13]. This way, learning can focus on higher-level goals. In our experiments, we apply our approach to four different tasks and show that the learned policies for a given task can be transferred to other road networks containing different SSRs. To conclude, the contributions of this paper are (1) the definition of non-deterministic spatial mobility tasks covering various spatial applications, (2) a reinforcement learning approach based on the options framework that combines dedicated routing algorithms with function approximation based policy learning, and (3) a neural function approximation which learns policies which are transferable between road networks.

2 Related Work

There are various approaches to solve spatio-temporal problems with Machine Learning and reinforcement learning methods. The works in [7, 8, 9, 17] apply reinforcement learning (RL) to solve the Travelling Salesman Problem (TSP), or other variants of the Vehicle Routing Problem (VRP). In contrast to these works, we do not decide on the optimal order in which to visit a given set of locations but find policies that determine the next location an agent should go to, given the current state of the environment. A related spatial task is the travelling officer problem (TOP) first described by [27]. In this task, the agent (the officer) tries to maximize the number of parking violations fined within a certain time frame. Shao et al. claim that the optimal solution is a linear program optimising the agent's paths [27, 26]. In contrast, we argue that parking violations fall under the category of stochastic spatial resources (SSRs), such that the optimal solution for TOP, among other tasks, should rather be defined as a (semi–) Markov decision process (SMDP).

The task of resource routing (RR) [24] aims at finding a single available resource such as a parking bay. In their work, the stochastic behaviour of resources is modelled by a continuous-time Markov chain and the proposed MDP is solved with real-time dynamic programming. In the follow-up work [23], approximate planners were proposed to scale up the solution to settings with larger numbers of resources. In our work, we also consider spatial tasks as MDPs. However, our new approach does not rely on any model assumption w.r.t. the underlying transition probabilities. Instead, we employ model-free reinforcement learning to solve the problem.

Besides TSP, popular applications to which spatial RL techniques have been applied successfully include fleet-management tasks like the taxi-dispatching problem (TDP) [10, 12, 30] or ride-sharing tasks [11, 1]. All of these approaches, however, work on simplified state spaces based on grid cells or spatial areas and often also aggregate large numbers of time steps into intervals of several minutes to make solutions tractable. In contrast, our new approach works directly on the road network representing the spatial environment.

The temporal abstraction used in this paper falls into the framework of hierarchical reinforcement learning (HRL) which has been regarded as essential for learning complex tasks early on [4]. In HRL, a task is decomposed into several subtasks that can be solved individually with a partial policy for each subtask. HRL is more efficient than trying to solve the entire problem at once and can speed up the learning process significantly by decreasing the number of impractical actions. In the past two decades, several hierarchical RL models, such as hierarchies of abstract machines (HAM) [18] and MAXQ value function decomposition [5], have been proposed. Many of these approaches are based on pre-defined subgoals and auxiliary pseudo-rewards. In addition, there has been put quite some effort into learning useful subgoals and reusable sub-policies [14, 31]. A general and very commonly used framework that has been defined independently of any assumption about subtasks or -goals is the options framework introduced by Sutton [29]. In this model, an option simply denotes a partial policy that can terminate at any time depending on the current state. The authors show that adding options to the underlying MDP conveniently results in an SMDP. In [2], the authors present the option-critic architecture that learns the partial policies (called option policies) along with the termination conditions and the policy for choosing the options (called policy over options) without the need to provide any subgoals or additional rewards. The tasks in this paper involve the subtask of routing to a certain node (intersection) for which efficient planning like Dijkstra exist. Thus, we base our work on the options framework and do not consider defining additional subgoals manually. We will compare our solution to the option-critic implementation which is the HRL approach that comes closest to ours. Temporal abstractions based on options have also been used in the context of spatial RL by [30]. However, [30] focuses only on the particular task of taxi dispatching in a multi-agent environment and works on hexagonal grid cells instead of the nodes of the underlying street graph. To summarise, we distinguish from existing approaches by providing a general framework that can be applied to a wide range of applications. Also, our new approach is transferable within the same domain to previously unseen spatial areas.

3 Background

In a Markov decision process (MDP), an agent observes an environment's state s, decides on an admissible action a, and subsequently receives an immediate reward $R(s, a)$ at each time step t. To allow for variable action lengths, the more general framework of semi-Markov decision processes (SMDPs), models the time between two successive decisions as a random variable τ.

Formally, a discrete-time semi-Markov decision process (SMDP) [6] is defined as a five tuple (S, A, R, P, γ), in which S is a set of states and A is a set of actions, with $A(s)$ denoting the set of admissible actions in a given state $s \in S$. The transition function $P(s', \tau \mid s, a)$

defines the probability that an agent traverses from state $s \in S$ to $s' \in S$ after taking action $a \in A(s)$ in $\tau \in \mathbb{N}^+$ time steps. The reward function $R(s,a) \to \mathbb{R}$ is a function that defines the expected reward for being in state $s \in S$, taking action $a \in A(s)$. We denote $r_{t+1}, r_{t+2}, \dots r_{t+\tau}$ as a sample of the distribution underlying R if taking an action took exactly τ time-steps. The discount factor γ is a value between zero and one and defines the optimization horizon of an agent.

Reinforcement learning (RL) provides a set of solvers for Markov decision processes (including SMDPs), without the need to know the exact MDP definition. Instead, an agent interacts with an environment, which contains the dynamics of the MDP. A commonly used technique is Q-Learning [32]. In this paper, we base our solution on Q-Learning. We aim to find a policy $\pi(a \mid s)$, i.e. a distribution of actions $a \in A(s)$, that maximizes the future expected rewards $V(s)$ for all states $s \in S$. The Q-Learning approach updates the Q-values (state-action values), $Q(s,a)$, by running trials in the environment. One can show that once the state-action values converged to the optimum, following the greedy policy is optimal:

$$\pi(a \mid s) = \begin{cases} 1 & \text{if } \arg\max_{a' \in A(s)} Q(s,a') = a \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The update rule for the Q-values after executing one action is given by:

$$Q(s_t, a) \quad \leftarrow \quad Q(s,a) \quad + \quad \alpha \left(\sum_{i=0}^{\tau-1} \gamma^i r_{t+i+1} + \gamma^\tau \max_{a' \in A(s_{t+1})} Q(s_{t+1}, a') - Q(s,a) \right) \tag{2}$$

When using the epsilon greedy policy, i.e. taking a random action with probability ϵ and otherwise the action of the greedy policy (Equation 1), one can show that the Q-values converge to the optimal values after an infinite number of trials. However, as soon as function approximations are used, e.g. [16], there are no theoretical convergence guarantees [3]. Nevertheless, in practice, it can usually be achieved that Q-learning converges to local optima.

3.1 Options

An option is a higher level temporal abstraction of action. That is to say, an option executes multiple actions until it decides to terminate. Hence, an option o consists of $(\mathscr{S}, \mu, \beta)$, where $\mathscr{S} \subseteq S$ is the set of possible states where the option can be started, μ is the policy defined for this option and $\beta : \mathscr{S} \to [0,1]$ is a function that defines the probability that this option terminates in a given state. We assume that any state where the option may continue is in $\mathscr{S}$. Every action $a \in A$ can be defined as a *one-step option* by defining $\mathscr{S} := \{s : a \in A(s)\}$, $\mu(a|s) = 1$, and $\beta(s) = 1$ for all states $s \in S$. For more details refer to [4].

Finding the optimal policy over options, i.e. $\pi(o \mid s)$, for all options $o \in O$ in all states $s \in S$, fits again into the SMDP framework [20, 29, 19, 21, 22]. However, in this case, one would treat options as opaque indivisible units, yet it is more powerful to look inside the options [20]. With intra-option learning methods, it is possible to update options after one action. Consequently, if the option policies are Markov, it is possible to update multiple options simultaneously, namely all options whose policies have the same action distribution in the given state [20]. If the distribution is not exactly equal, one may use importance sampling. The resulting update rule for one-step intra-option Q-Learning based on SMDPs is:

$$Q(s_t, o) \quad \leftarrow \quad Q(s_t, o) \quad + \quad \alpha \left(\sum_{i=0}^{\tau-1} \gamma^i r_{t+i+1} + \gamma^\tau V(s_{t+\tau}, o) - Q(s_t, o) \right), \tag{3}$$

where τ is the amount of time steps consumed by the current action. We define $V(s, o)$ as follows:

$$V(s, o) := (1 - \beta(s))Q(s, o) + \beta(s) \max_{o' \in O} Q(s, o') \tag{4}$$

It is possible to define hand-crafted options. Nevertheless, the usual target of general RL is to learn good options either directly [2], or by finding good sub-goals [15, 14, 31].

The authors of [29] showed that it is an improvement to the policy if you allow the option to interrupt before it terminates, to favour a better option. This is called *option interrupt*.

3.2 Non-Deterministic Spatial Mobility Tasks

The environment of a non-deterministic spatial mobility task (NSMT) consists of a (directed) road network $G = (N, E, t_{travel})$, where the nodes N represent crossings, the edges E represent road segments and $t_{travel}(e)$ represents the travel time for traversing segment e. Additionally the environment has a set of stochastic spatial resources (SSRs) $\mathcal{R}$ where each SSR $\rho \in \mathcal{R}$ has a location $n_\rho \in N$, a type ι_r (e.g. parking spot or destination) and a status vector x_ρ. x_ρ is task-specific and encodes information which is used to indicate the reward and the impact on the agent when arriving at the location of the SSR. For example, x_ρ might contain the amount of the reward and the time claiming the reward might take when an agent arrives at ρ. The location of the agent is denoted as $n_a \in N$. Besides the location, an agent may have further domain-specific features ζ (e.g. the remaining range of the vehicle). The action set A is defined as $A := E \times \{True, False\}$, where each edge $e = (n_i, n_j)$ with $n_i, n_j \in N$ can only be executed if $n_a = n_i$. Since the agent might not want to take every resource, the second part of the action indicates whether a resource should be consumed (if possible) or not. Executing an action increases the real-world time ϑ by

$t_{travel}(e)$ which can be transferred into discrete time steps by division with a multiple of the greatest common divisor of all edge travel times in the graph G. Let us note that x_ρ and n_ρ change over time and can be modified by the agent. Thus, SSRs can show arbitrary behaviour allowing for modelling a plethora of tasks. Furthermore, we assume that the behaviour of SSRs is non-deterministic and described by an unknown statistical process.

In the following, we will introduce four NSMTs which are used in the evaluation. In **resource routing** (RR) [24], the SSR state contains a binary variable indicating SSR availability. For the application of parking search, this means a parking bay is either available or occupied. The actual destination of the driver is modelled by another type of SSR, such that the agent can learn to prefer parking spots close to the destination. The search terminates after any SSR is consumed and to ensure that policies with shorter search times are preferred, we have to choose $\gamma < 1$. In **resource collection** (RC) the agent is rewarded for any available resource it collects within a finite horizon. The travelling officer problem [27] and taxi dispatching are instances of the resource collection task as parking offenders or taxi customers might leave before the agent's arrival. For the **chaser task**, we consider a single SSR which is always available but with a location varying over time. The agent receives a reward when getting close enough to the SSR which also ends the episode. As for RR, we choose $\gamma < 1$ to prefer shorter episodes. Applications involve intercepting suspicious vehicles, hunting an eloped cat equipped with GPS, or reaching a moving ice cream seller. In the **courier task**, there are two types of SSRs which appear pairwise: pickup SSRs and drop off SSRs. Consuming a pickup SSR does not grant a reward but changes the state of the corresponding drop off SSR to consumable for a given time interval. Rewards are earned by consuming these drop off SSRs. Episodes are finite horizon and agents should maximize the number of drop-offs. The courier task has applications in routing bike couriers or ride-sharing tasks where the same agent can work on several deliveries in parallel. Let us note that these settings are just a small selection of tasks where our method works and more complicated NSMTs could be formulated by considering more SSR and agent variables.

4 Solution

Target Options. The idea behind our approach, named ORIENTATION, is to separate the task of deciding which location should be visited to positively affect future rewards from the task of planning a route to get there. We argue that in many settings this navigation subtask can be solved by efficient deterministic planers such as Dijkstra's algorithm, whereas the decision about where to navigate next is more difficult due to the non-deterministic nature of the SSRs. To integrate this idea into reinforcement learning, we employ the options framework as described above. We define the set of all options as follows: $O := \{o_n \mid n \in N\}$, where $o_n := (\mathscr{S}_n, \mu_n, \beta_n)$ is an option whose policy μ_n follows the shortest path to node $n \in N$ from the position n_a of the current state and potentially consumes resources only at the destination. Further, let us define the set of input nodes $\mathscr{S}_n := N \setminus \{n\}$. The option terminates *iff* the node of the current state is the target node n, i.e. $\beta_n(s)$ is 1 if the current state's node n_a is n and 0 otherwise. Note that we do not need any one-step options, since there is always at least one option for each edge, i.e. the option for the target node of the edge. Usually, the amount of options $|O|$ is smaller than the amount of actions $|A|$, since in connected graphs $|E| \geq |N| - 1$. The policy of an option is computed of an arbitrary routing algorithm computing the shortest path from n_a to the target node n of option o_n. For simplicity, we employed Dijkstra's algorithm in our experiments. Nevertheless, more efficient methods could be employed to save time for larger road networks.

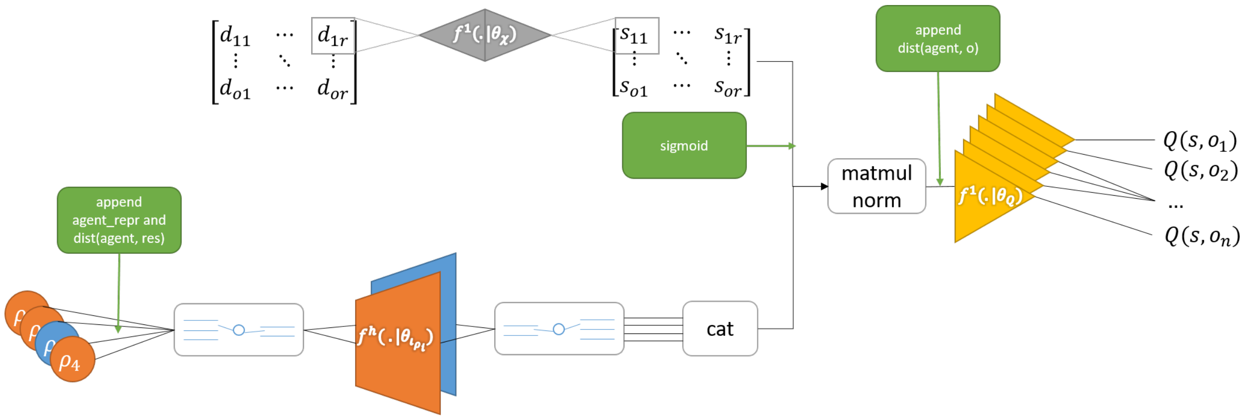


Figure 1: Illustration of the function approximation.

Function Approximation. In this section, we explain our proposed neural network architecture to approximate the Q-function. In the following, we denote $f^h(\cdot|\theta)$ as a standard multilayer perceptron (MLP) with h output neurons, the parameter set θ and unless otherwise stated a ReLU unit as non-linearity between both dense layers. Furthermore, the distance function $dist(n_i, n_j)$ describes the length of the shortest paths between n_i and n_j in the network G w.r.t. travel time. The first step of our function is to convert each SSR $\rho_i \in \mathcal{R}$ to a feature representation $x_i := f^h([\rho_i, dist(n_a, n_{\rho_i}), \zeta] \mid \theta_{\iota_{\rho_i}})$, where n_a is the location of the agent and n_{ρ_i} is location of the SSR ρ_i. $\theta_{\iota_{\rho_i}}$ is the parameter set for the feature representation shared along all SSRs $\rho_i \in \mathcal{R}$ with the same type ι_{ρ_i}. Note that this step enables the use of differently shaped resource representations. Next, we want to compute an embedding for each option $o_j \in O$. We propose to use a distance based weighting function $\chi(o_j, \rho_i) := sigmoid(f^1(dist(n_j, n_i) \mid \theta_\chi))$, where θ_χ are the parameters for the weighting function. The non-linearity in this multilayer perceptron is the sigmoid function. χ determines the relevance of a stochastic element to the embedding of option o_i and is not necessarily a function of the distance between the option target and the location of the SSR. For instance, one could replace it with an attention mechanism, where

```
$\pi^\theta \leftarrow$ epsilon greedy policy w.r.t. $Q(s, o \mid \theta)$
exp_buffer $\leftarrow$ queue(maxlen)
initialize $\theta$ and $\hat{\theta}$ randomly
for episode in #episodes do
    done $\leftarrow$ false
    while not done do
        $s, a, r, s', done, \tau \leftarrow act(\pi^\theta)$
        exp_buffer.add($(s, a, r, s', done, \tau)$)
        if every $k^{th}$ step ($k = 4$) then
            minibatch = sample(exp_buffer)
            for $s_i, a_i, r_i, s'_i, done_i, \tau_i \in minibatch$ do
                choose set of options $\mathcal{O}_i$ where next edge on the shortest path from
                  the node defined by $s_i$ to target node of the option is $a_i$
                for $o_j \in \mathcal{O}_i$ do
                    if $done_i$ then
                        $y_{i,j} \leftarrow r_i$
                    else
                        $y_{i,j} \leftarrow$
                          $r_i + \gamma^{\tau_i} \cdot \Big((1 - \beta(s'_i))Q(s'_i, o_j \mid \hat{\theta}) + \beta(s'_i) \max_{o' \in O} Q(s'_i, o' \mid \hat{\theta})\Big)$
                    end
                end
            end
            update $\theta$ w.r.t. loss $(y_{ij} - Q(s_i, o_j))^2$
            if every $l^{th}$ time then
                $\hat{\theta} \leftarrow \theta$
            end
        end
    end
end
return $\pi^\theta$
```

Algorithm 1: Pseudocode of ORIENTATION

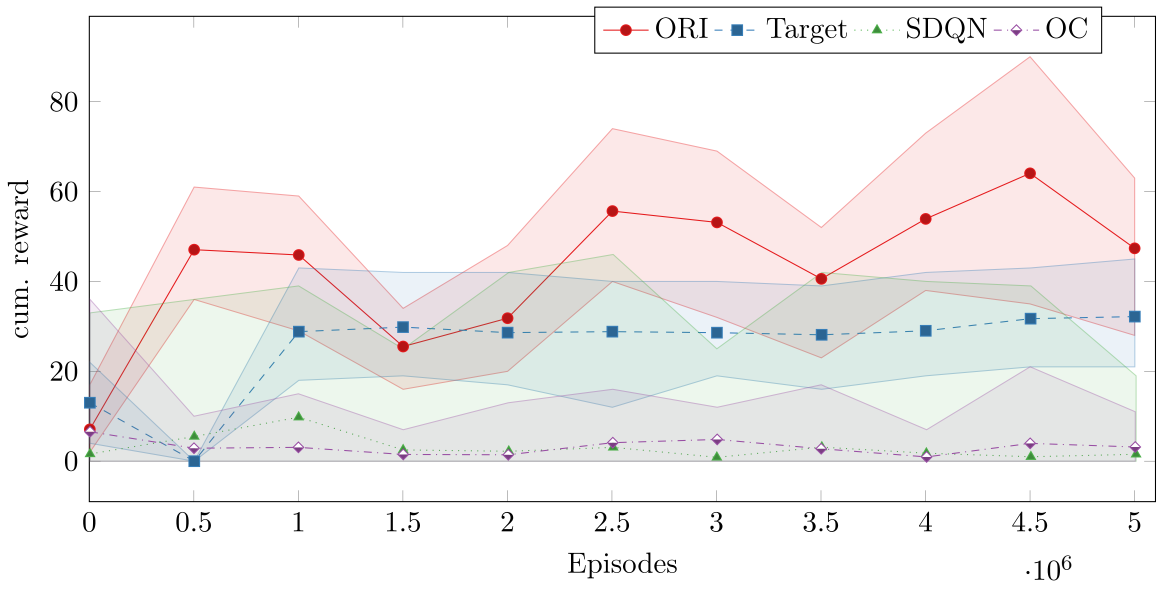


Figure 2: Test performance in the Resource Collection setting for Kraiburg (KB). The y-axis represents the travel time until the opponent got caught.

the key is any kind of option representation (e.g. target node representation) and the values are the x_i representations for all SSRs. However, the distance-based weighting worked well in our settings and is transferable. Hence, we compute the target option representation τ_j for option o_j as follows: $\tau_j = {}^1/_{|\mathcal{R}|} \cdot \sum_{\rho_i \in \mathcal{R}} \chi(o_j, \rho_i) \cdot x_i$. In the next step, we compute $Q(s, o_j) := f^1([\tau_j, dist(n_a, n_j)] \mid \theta_Q)$, where θ_Q is the respective parameter set. Let us note that $f^1([\tau_j, dist(n_a, n_j)] \mid \theta_Q)$ receives a concatenation of option description τ_j in addition to the distance between the agent location n_a to the option target node n_j to integrate the information about the time required to follow option o_j. Figure 1 provides an overview on the complete architecture. The pseudocode can be found in Algorithm 1.

Efficient Path Calculations. For computing τ_j, we need the shortest path distances from every stochastic element position to every other node (option targets). In small graphs, the all pair shortest paths can be pre-computed to speed up training. One might argue that this becomes infeasible very fast. However, learning a spatial task on street graphs being larger than the feasible size for pre-computing is usually intractable with standard reinforcement learning approaches. Nevertheless, if one wants to train or apply it to a very large graph, computing a (truncated) reversed Dijkstra from the stochastic elements $\rho \in \mathcal{R}$ can be performed if $|\mathcal{R}| << |N|$. As one can assume that far off stochastic elements do not have any effect on the target options, one might truncate the reversed Dijkstra algorithms at some maximum distance. Yet, this is task-specific. In Orientation, we need to efficiently retrieve all options whose policy would take the given action in a given state. A naive algorithm would compute the shortest path to all nodes and checks if the next edge is equal to the edge defined by the action. This is very inefficient and we can do better. Our proposal is to store all shortest paths in a matrix $\mathcal{P}$, where $\mathcal{P}_{ij}$ is the id for the next node when the source node is the i^{th} node in N and the target node is the j^{th} node in N of the shortest path. Now, we want to retrieve all target nodes whose shortest path from node n_a traverse the edge $e_a \in E$ defined by the given action a. Therefore, we check which entries of the row $\mathcal{P}_k$ are equal to the destination node n_d of e_a, where k is the id of the given state's node n_a. Hence, we simply compute $\mathcal{P}_i == n_d$, which can be efficiently calculated on GPU with state-of-the-art deep learning libraries.

5 Environments

In this section, we present the environments used in our experiments. All environments contain a graph that is either synthetically generated (grid graph) or downloaded from OpenStreetMap[1]. Further, they all have a step function that is moving the agent's position along a (valid) edge from one node to another. Thereby, the agent consumes time. In this step function, the agent can further decide whether it wants to consume a resource (if possible). The step function returns the next state, reward and consumed time. This is task specific. Hence, every task extends the step function with additional logic, as described below.

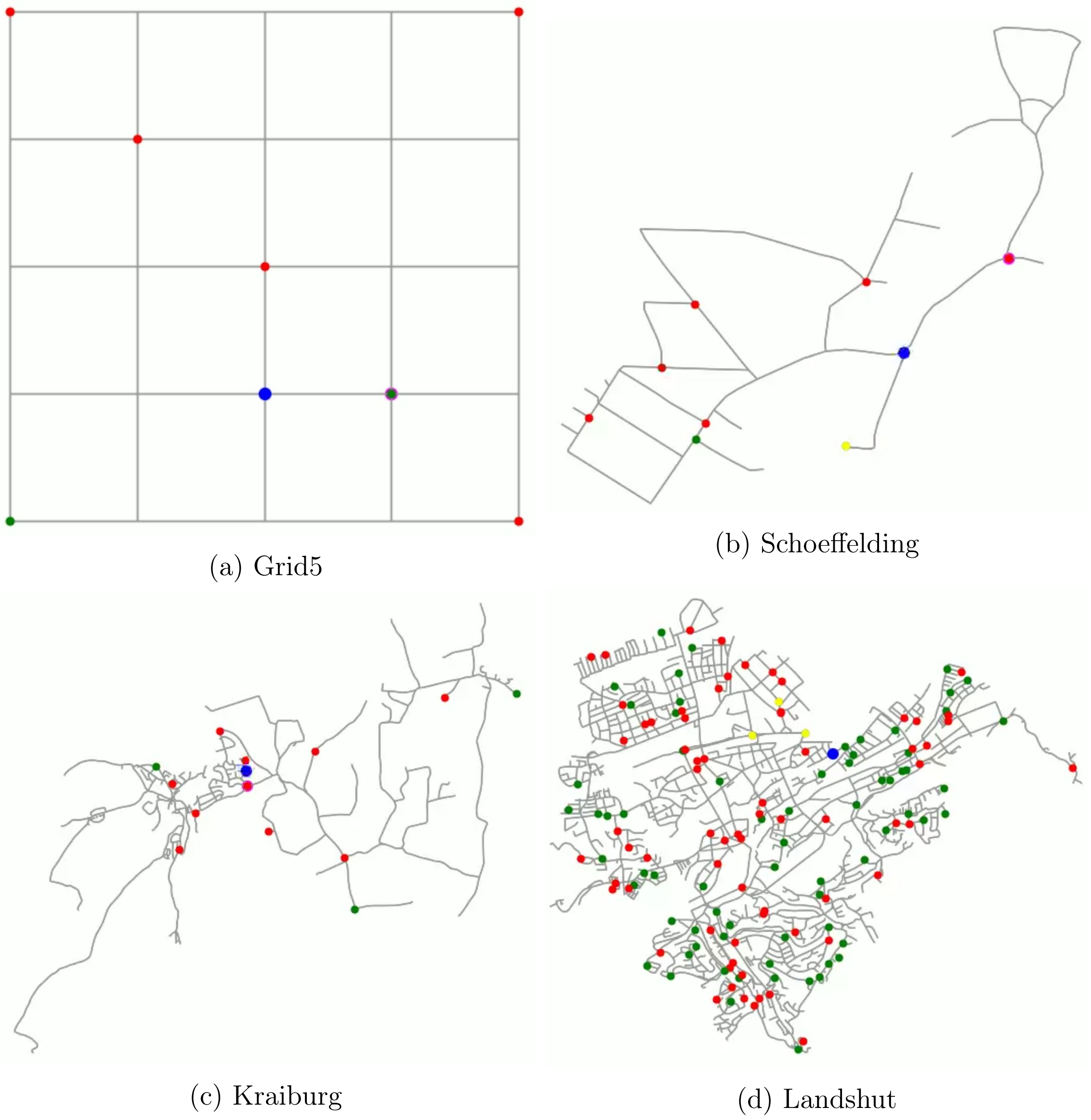
Resource Routing. In the *resource routing* task, the goal of the agent is to route to the destination as quickly as possible. However, the agent needs to acquire a resource before it can walk to the destination. After claiming the resource, the agent possibly changes its mode of transportation, i.e. his travel speed and directly traverses to the destination node. For instance, the agent could represent a driver searching for a parking spot. There are multiple resources in this environment. However, resource availability is stochastic since they might become occupied in the near future. The actual destination is represented as an SSR with a different type than the parking spots. That way, the agent can learn to prefer parking spots close to the destination. For simulating a resource routing (RR) task, we model the residence times of parking spot states (available/occupied) with a Poisson process and sample residences time from an exponential distribution. The locations of the resources are fixed though (parking spots do not move). The agent receives a positive reward of 1 if it reaches the destination after claiming a parking spot and walking to the destination, otherwise the reward is 0. The agent representation ζ is always 1. The episode ends when the agent successfully claims a resource or a predefined time horizon is reached.

[1]`https://www.openstreetmap.org/`

Resource Collection. In the *resource collection* task, the agent tries to claim as many available resources as possible within a certain time frame. As in the resource routing task, there are multiple resources with fixed locations but stochastic availabilities. Our implementation models them as a Poisson process and sample residence times from an exponential distribution. By contrast to the other tasks described above, here, an episode does not end after a resource is acquired, but continues until a certain fixed time has passed (horizon). An available resource may be collected only once. Then, it changes its state to collected, unless it re-appears. The SSR representation x_ρ consists of two boolean, i.e. the availability and if it has been visited/collected already. In this implementation, the agent representation ζ is always 1, but it could e.g. be the remaining horizon if the stochastic process changes over time.

Chaser. To also model moving targets, in the *chaser* task, the agent has to "catch" another agent that performs a random walk on the street graph and moves ten percent slower. The target agent is caught if the chasing agent traverses an adjacent edge at the same tick of the simulation. The agent then receives a reward of 1 and the episode ends. In contrast to resource routing, there is only one SSR with a fixed state, i.e. $x_\rho = 1$ (it is always available), however its location is stochastic/dynamic. The agent representation ζ is always 1. The episode ends when the agent successfully claims a resource or a predefined time horizon is reached.

Courier. In the *courier* task, a deliverer tries to execute as many delivery requests as possible. A request consists of a start location and a destination. Real world examples of this task are food delivery or a bike messenger delivering urgent mails. In the courier task, there are three different kinds of SSRs, namely the start and destination of new requests and the destination of packages already picked up. The representation x_ρ of a start SSR is the one-vector and the x_ρ of a destination is the one-hot encoding of its request's state (already picked up or not). The agent receives a positive reward (+1) if it successfully executed the delivery. Withal, the agent may pick up multiple requests before delivering. However, if the agent is not able to deliver the package in a certain amount of time, the customer is not interested in the delivery anymore. E.g. the customer might say, if the pizza is not delivered in two hours, he/she does not want any pizza anymore. For simplicity, in our experiments, the start and destination nodes are sampled uniformly. Due to the varying number of resources (and hence input size) over time, the baseline function approximations are not applicable anymore in this task. However, our proposed function approximation can handle this issue out of the box.


(a) Grid5

(b) Schoeffelding

(c) Kraiburg

(d) Landshut

Figure 3: Illustration of the graphs used in the experiments (here: resource collection). The red/green dots denote (non-)collectable resources. The blue dot is the agent's position. Yellow dots denote recently collected resources.

6 Experiments

For our experiments, we implemented simulation environments for the four tasks mentiond above: resource routing, resource collection, chaser and courier. All environments contain a graph that is either synthetically generated (grid graph) or downloaded from OpenStreetMap. We use the small street network of the German village Schöffelding (**Schoeff**) with 40 nodes and 92 edges to train our approach and two larger graphs, namely Kraiburg (**KB**) with 249 nodes and 578 edges and Landshut with 1422 nodes 3503 edges (see Figure 3). Resource availability times are modelled by a Poisson process. Environments and agents are coded in Python (Pytorch) and training is run on a GTX 1080 or stronger GPU.

We compare our approach, ORIENTATION (abbreviated in the tables as *ORI*), with four other deep reinforcement learning baselines on the four different spatial applications described above. The baselines are described below:

The first baseline approach is a standard DQN [16] adopted for SMDPs. The function approximation learns node embeddings for the source and target node, concatenates them and applies a standard multi-layer perceptron on top of the concatenated embeddings. The possible actions at each node include only the neighboring nodes to which an outgoing edge exists (primitive actions). We call this baseline *RDQN* (for Routing DQN).

The next baseline is similar to the RDQN, as it is a standard DQN adapted for SMDPs. However, its state representation and function approximation is applicable to all NSMTs except the courier task. We call it *SDQN*. The function approximation learns a node embedding for the agent's current position, and uses the same multi-layer perceptron for all $\rho \in \mathcal{R}$ with x_ρ^t as input. All representations are concatenated and a multi-layer perceptron predicts the Q-values for all primitive actions.

The *Target* baseline shows how important the proposed function approximation is. Target is equal to the ORIENTATION approach, except that the function approximation has been replaced by the function approximation as described in the SDQN. Thus, the primitive actions are extended by the target options to all nodes.

OC (option-critic) is a famous framework for learning options end-to-end [2]. In contrast to our approach, it learns the option policies of 20 options as well, i.e., it does not make use of standard routing algorithms. The policy over options is trained with the SMDP baseline. All one-step options are also available.

In each experiment, we continuously evaluate the different approaches after a fixed number of episodes has passed by stopping the training and calculating the averages of the objective over 100 different episodes. The Plots additionally show the value range (min/max values) shaded in the same color.

Resource Routing and Chaser. Table 1 displays the discounted cumulative reward on three different graphs for the resource routing and chaser tasks. Let us note that a higher discounted reward corresponds to shorter search time as the γ was set to $0.01^{1/h}$ where

Task	Graph	Steps	Approach	Objective
RR	Grid5	500k	ORI	0.891
			SDQN	**0.901**
			Target	0.894
			OC	0.296
	Schoeff	500k	ORI	0.553
			SDQN	0.711
			Target	0.412
			OC	0.589
	KB	1M	ORI	**0.315**
			SDQN	0.091
			Target	0.065
			OC	0.002
Chaser	Grid5	500k	ORI	0.932
			SDQN	0.935
			RDQN	0.897
			Target	**0.938**
			OC	0.650
	Schoeff	500k	ORI	0.643
			SDQN	**0.715**
			RDQN	0.513
			Target	0.635
			OC	0.274
	KB	500k	ORI	**0.714**
			SDQN	0.099
			RDQN	0.093
			Target	0.405
			OC	0.105

Table 1: Performance for Resource Routing and Chaser task. Objective is the mean discounted cumulative reward.

h is the horizon. For RR, we consider 5, 40 and 7 SSRs for the Grid5, Schoeff and KB networks. Results indicate that standard methods perform well on the small networks but are outperformed by ORIENTATION for the larger KB network. For RR on the KB network, ORIENTATION was the only method finding a working policy even though all networks were trained for 1 million steps which are twice as long as for the other settings. To conclude, in smaller settings and in particular, in the simple grid graph all methods offered viable policies and ORIENTATION sometimes falls behind the compared methods. This indicates that all

Task	Graph	Steps	Approach	Objective
RC	Grid5	500k	ORI	**56.39**
			SDQN	34.04
			Target	36.30
			OC	7.91
	Schoeff	500k	ORI	**33.7**
			SDQN	23.0
			Target	18.37
			OC	22.2
	KB	5M	ORI	**48.27**
			SDQN	1.75
			Target	32.11
			OC	1.74
Courier	Grid5	500k	ORI	38.9
		0	Requests	46.8
	Schoeff	4M	ORI	34.93
		0	Requests	36

Table 2: Performance for Resource Collection and Courier task. Objective is the mean cumulative reward.

methods are suitable for solving the task as long as networks are small. However, when considering a larger setting as the KB graph, we can observe that the standard methods do not offer viable policies in most cases and ORIENTATION outperforms all comparison partners.

Resource Collection. For resource collection (RC), we report the cumulative reward corresponding to the number of consumed SSRs. We consider 9, 40 and 13 SSRs for the Grid5, Schoeff and KB networks. Results can be seen in table 2. For this setting, long term planning is more important and thus, ORIENTATION outperforms all other methods on all three graphs. Again we can observe that only the methods using the target options Target and ORIENTATION managed to learn a viable policy for the larger KB graph. Let us note that we had to train all approaches with 5 million steps for the KB graph as the setting is far more challenging. Figure 2 shows the discounted cumulative reward on KB. As can be seen, OC and SDQN are not capable to learn a viable policy whereas ORIENTATION quickly learns how to improve the number of collected resources. Only the Target approach offers a viable policy as well. However, the performance gap between Target and ORIENTATION demonstrates the effectiveness of our proposed function approximation.

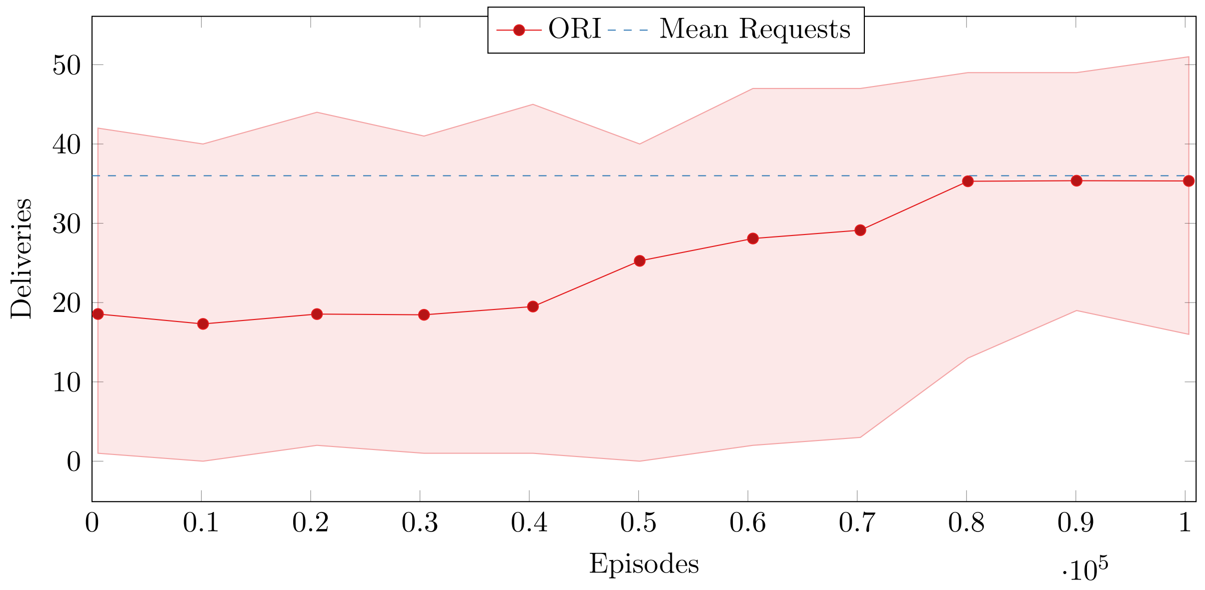


Figure 4: Test performance on the Schöffelding graph w.r.t. training episodes. The dashed blue line indicates the mean of available requests per episode. Hence, it is an upper bound.

Courier Task. The courier task is different from the other three tasks as the number of SSRs cannot be predetermined as new delivery requests spawn randomly. However, all three baseline approaches require a fixed number of SSRs for their function approximation. In contrast, ORIENTATION can handle these circumstances as it aggregates the currently present SSRs early on in the function approximation. Since we could not apply our comparison partners here, we compare our results to the expected number of delivery requests spawning during an episode as this yield an upper bound for the performance. In table 2, we display the cumulated reward of ORIENTATION compared to the average number of available request in each episode. As can be seen, ORIENTATION displays a viable result on the simple grid graph and comes very close to an optimal policy on the Schoeff network. Furthermore, figure 4 displays the test performance of ORIENTATION during training indicating a constant increase of served requests.

Transfer between Networks. To demonstrate that the policies learnt by ORIENTATION offer viable solutions when applied to a different network, we applied the policy, we learned on the grid graph to the road networks of Schoeffelding (Schoeff) and Kraiburg (KB). Tables 3, 4, 5 and 6 display the results measured in discounted cumulative rewards for RR and chaser and cumulative reward for the RC and courier tasks. As can be seen, in most cases the results of the transferred neural network weights are competitive with the results of the agents trained for the target street network. However, if the implicit model assumptions

Target Graph \Source Graph	Grid5	Kraiburg	Landshut	Schoeffelding
Grid5	0.891	0.769	0.229	0.000
Kraiburg	0.250	0.173	0.000	0.000
Landshut	0.609	0.000	0.587	0.025
Schoeffelding	0.497	0.471	0.615	0.553

Table 3: Transfer results for the Resource Routing environment in cumulative discounted rewards.

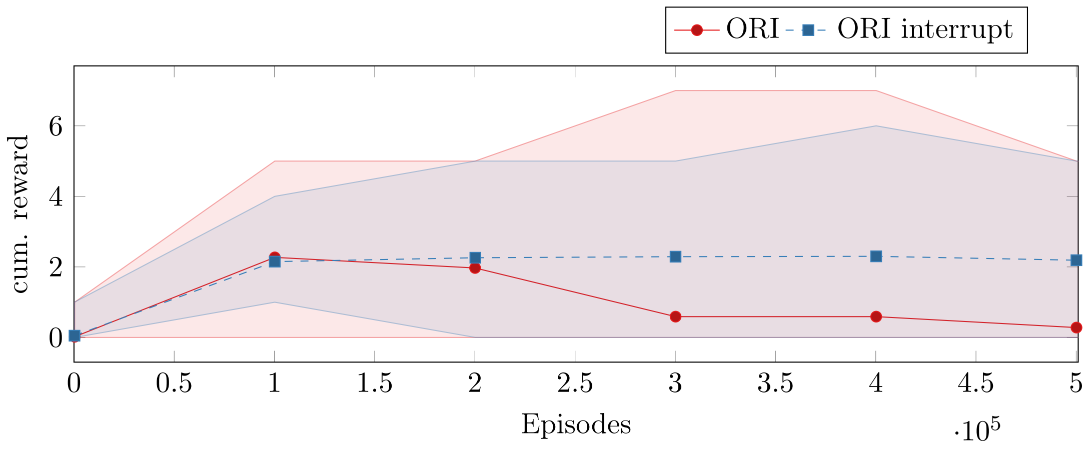
Target Graph \Source Graph	Grid5	Kraiburg	Landshut	Schoeffelding
Grid5	0.931	0.788	0.923	0.781
Kraiburg	0.729	0.360	0.734	0.253
Landshut	0.686	0.101	0.501	0.110
Schoeffelding	0.697	0.467	0.727	0.436

Table 4: Transfer results for the Chaser environment in cumulative discounted rewards.

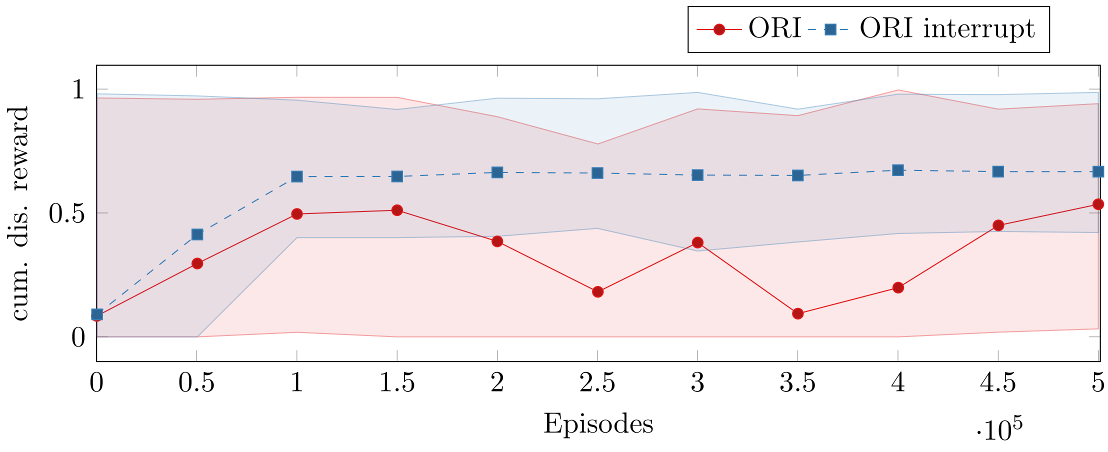
differ too much from the environment model, the agent can perform poorly. For instance, if the resources' availability changes in a very different frequency than expected, the actions may become sub-optimal. On the other hand, the weights trained for the small graphs sometimes significantly outperform the agent trained directly for the target graph. Two reasons may cause this effect. First, the dynamics of the resources are similar and the distances between the resources have the same magnitude. Second, the amount of training steps is not enough to let the agent fully converge close to the optimum at the target network in the given environment. Nevertheless, it seems to be enough for the smaller Grid5 setting. This suggests that training a task on a smaller graph and fine-tune the weights one the target graph is possible to decrease training time.

Target Graph \Source Graph	Grid5	Kraiburg	Landshut	Schoeffelding
Grid5	56.31	12.57	42.72	0.00
Kraiburg	16.02	7.02	2.49	0.00
Landshut	1.19	0.91	0.62	0.04
Schoeffelding	111.17	19.54	20.51	34.71

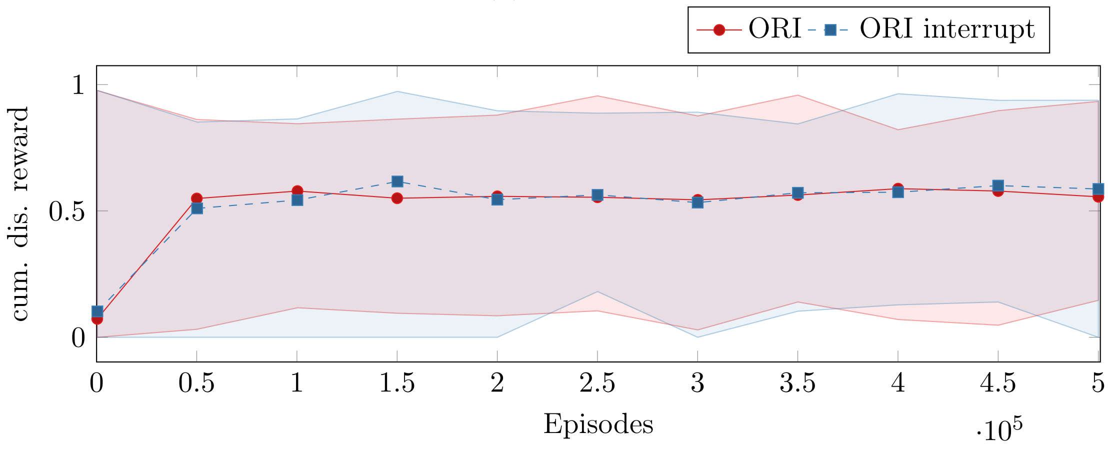
Table 5: Transfer results for the Resource Collection environment in cumulative rewards (number of collected resources).



(a) Landshut RC.



(b) Landshut Chaser.



(c) Landshut RR.

Figure 5: Effect of option interrupt in Landshut.

Target Graph \Source Graph	Grid5	Landshut	Schoeffelding
Grid5	32.56	0.30	0.22
Kraiburg	1.06	0.00	0.00
Landshut	1.88	0.00	0.00
Schoeffelding	30.42	0.08	0.08

Table 6: Transfer results for the Courier environment in cumulative rewards.

Option Interrupt. So far, we have not made use of the option interrupt, i.e. stopping one option for executing another option with better Q-value. In Figure 5, we illustrate the effect of option interrupt on the Landshut graph. As one can observe, the policy converges more stable and to better results in the chaser and RC task. At the RR task, both policies are comparable. On small graphs, we did not observe much improvement, which is why we omitted option interrupt in the settings before. However, if the distances become larger, reacting on unexpected changes in the state becomes more important.

7 Conclusion & Future Work

Non-deterministic spatial mobility tasks (NSMTs) describe settings where an agent traverses a road network and receives a reward when reaching a stochastic spatial resource (SSR) having a suitable SSR state. The SSR state depends on an unknown stochastic process and in addition, agent actions can also modify SSR states. As the non-deterministic behaviour of SSRs often results in an exponentially growing state space, reinforcement learning is a suitable way to find efficient policies for NSMTs. However, common approaches either do not scale with the road network or aggregate the spatial environment into grid cells. In this paper, we present a scalable framework for solving NSMTs on node-level. To let learning focus on higher-level goals, we introduced a set of "target options" for routing to any node in the graph that makes use of efficient deterministic routing algorithms. Moreover, we propose a sample efficient function approximation that shares parameters overall dynamic spatial elements. In our experiments, we show that our approach can solve four different NSMTs and even show that policies can be transferred to different road networks. In the future, we plan to investigate how the number of calculated target options could be reduced. Moreover, we want to extend our approach to multi-agent settings, which are very common in spatial tasks.

References

[1] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *CoRR*, abs/1903.03882, 2019.

[2] P. Bacon, J. Harb, and D. Precup. The option-critic architecture. In S. P. Singh and S. Markovitch, editors, *AAAI*, pages 1726–1734. AAAI Press, 2017.

[3] L. Baird. Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning Proceedings 1995*, pages 30–37. Elsevier, 1995.

[4] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete event dynamic systems*, 13(1-2):41–77, 2003.

[5] T. G. Dietterich. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of artificial intelligence research*, 13:227–303, 2000.

[6] R. A. Howard. Dynamic probabilistic systems, volume 2: Semi-markov and decision processes, 1971.

[7] W. Joe and H. C. Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 394–402, 2020.

[8] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song. Learning combinatorial optimization algorithms over graphs. In *N(eur)IPS*, pages 6348–6358, 2017.

[9] W. Kool, H. van Hoof, and M. Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[10] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *WWW*, pages 983–994, 2019.

[11] M. Li, Z. Qin, Y. Jiao, Y. Yang, J. Wang, C. Wang, G. Wu, and J. Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *WWW*, 2019.

[12] K. Lin, R. Zhao, Z. Xu, and J. Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *SIGKDD*, page 1774–1783. ACM, 2018.

[13] A. Madkour, W. G. Aref, F. U. Rehman, M. A. Rahman, and S. M. Basalamah. A survey of shortest-path algorithms. *CoRR*, abs/1705.02044, 2017.

[14] A. McGovern and A. G. Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *ICML*, page 361–368. Morgan Kaufmann Publishers Inc., 2001.

[15] E. A. Mcgovern. *Autonomous discovery of temporal abstractions from interaction with an environment*. PhD thesis, University of Massachusetts at Amherst, 2002.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

[17] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pages 9839–9849, 2018.

[18] R. Parr and S. J. Russell. Reinforcement learning with hierarchies of machines. In *N(eur)IPS*, pages 1043–1049, 1998.

[19] R. E. Parr. *Hierarchical Control and Learning for Markov Decision Processes*. PhD thesis, University of California, 1998.

[20] D. Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts, 2000.

[21] D. Precup and R. S. Sutton. Multi-time models for temporally abstract planning. In *N(eur)IPS 10*, pages 1050–1056. MIT Press, 1998.

[22] D. Precup, R. S. Sutton, and S. Singh. Theoretical results on reinforcement learning with temporally abstract options. In *ECML*, pages 382–393. Springer, 1998.

[23] S. Schmoll, S. Friedl, and M. Schubert. Scaling the dynamic resource routing problem. In *SSTD*, pages 80–89, 2019.

[24] S. Schmoll and M. Schubert. Dynamic resource routing using real-time dynamic programming. In *IJCAI*, pages 4822–4828, 2018.

[25] S. Schmoll and M. Schubert. Vision paper: Reinforcement learning in smart spatio-temporal environments. In *ACM SIGSPATIAL*, page 81–84, 2018.

[26] W. Shao, F. D. Salim, J. Chan, S. Morrison, and F. Zambetta. Approximating optimisation solutions for travelling officer problem with customised deep learning network. *CoRR*, abs/1903.03348, 2019.

[27] W. Shao, F. D. Salim, T. Gu, N. Dinh, and J. Chan. Traveling officer problem: Managing car parking violations efficiently using sensor data. *IEEE Internet of Things Journal*, 5(2):802–810, 2018.

[28] M. M. Solomon and J. Desrosiers. Survey paper—time window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988.

[29] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

[30] X. Tang, Z. T. Qin, F. Zhang, Z. Wang, Z. Xu, Y. Ma, H. Zhu, and J. Ye. A deep value-network based approach for multi-driver order dispatching. In *SIGKDD*, page 1780–1790, 2019.

[31] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. *CoRR*, abs/1703.01161, 2017.

[32] C. J. C. H. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, May 1989.

A.7 Optimizing the Spatio-Temporal Resource Search Problem with Reinforcement Learning (GIS Cup)

Publication

Felix Borutta, Sebastian Schmoll, and Sabrina Friedl. Optimizing the spatio-temporal resource search problem with reinforcement learning (GIS cup). In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 628–631. ACM, 2019

DOI: `https://doi.org/10.1145/3347146.3363351`

Contribution

Felix Borutta developed a round-trip based agent for the GIS Cup 2019. Felix Borutta and Sabrina Friedl developed a statistical model for choosing the round-trips. Sebastian Schmoll proposed the multi-armed bandit approach that improves the statistical model.

A.8 SMART-Env

Publication

Sabrina Friedl, Sebastian Schmoll, Felix Borutta, and Matthias Schubert. Smart-env. In *21st IEEE International Conference on Mobile Data Management, MDM 2020, Versailles, France, June 30 - July 3, 2020*, pages 234–235. IEEE, 2020

DOI: `https://doi.org/10.1109/MDM48529.2020.00050`

Contribution

Sabrina Friedl and Felix Borutta developed the simulation core for the SMART-Env. Sabrina Friedl designed and implemented the interface for reinforcement learning agents. Sebastian Schmoll contributed with discussions. Sebastian Schmoll implemented the user interface for SMART-Env. Sabrina Friedl contributed with style improvements to the user interface. Sabrina Friedl mainly wrote the text for the publication.

Bibliography

[1] Abubakr O Al-Abbasi, Arnob Ghosh, and Vaneet Aggarwal. Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning. *IEEE Transactions on Intelligent Transportation Systems*, 20(12):4714–4727, 2019.

[2] Andrew G Barto, Steven J Bradtke, and Satinder P Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1):81–138, 1995.

[3] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[4] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[5] Felix Borutta, Sebastian Schmoll, and Sabrina Friedl. Optimizing the spatio-temporal resource search problem with reinforcement learning (GIS cup). In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 628–631. ACM, 2019.

[6] Di Chai, Leye Wang, and Qiang Yang. Bike flow prediction with multi-graph convolutional networks. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '18, page 397–400, New York, NY, USA, 2018. Association for Computing Machinery.

[7] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[8] Luca Foti, Jane Lin, and Ouri Wolfson. Optimum versus nash-equilibrium in taxi ridesharing. *GeoInformatica*, pages 1–29, 2019.

[9] Sabrina Friedl, Sebastian Schmoll, Felix Borutta, and Matthias Schubert. Smart-env. In *21st IEEE International Conference on Mobile Data Management, MDM 2020, Versailles, France, June 30 - July 3, 2020*, pages 234–235. IEEE, 2020.

[10] Kaiqun Fu, Taoran Ji, Liang Zhao, and Chang-Tien Lu. Titan: A spatiotemporal feature learning framework for traffic incident duration prediction. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, page 329–338, New York, NY, USA, 2019. Association for Computing Machinery.

[11] Daniel A Garcia-Ulloa, Li Xiong, and Vaidy Sunderam. Truth discovery for spatio-temporal events from crowdsourced data. *Proceedings of the VLDB Endowment*, 10(11):1562–1573, 2017.

[12] Hector Geffner. Model-free, model-based, and general intelligence. *arXiv preprint arXiv:1806.02308*, 2018.

[13] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, volume 5, pages 156–165, 2005.

[14] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[15] Jörg Hoffmann and Bernhard Nebel. The ff planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[16] Ronald A Howard. Dynamic programming and markov processes. 1960.

[17] Ronald A Howard. Dynamic probabilistic systems, volume 2: Semi-markov and decision processes, 1971.

[18] Waldy Joe and Hoong Chuin Lau. Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 30, pages 394–402, 2020.

[19] Gregor Jossé, Klaus Arthur Schmid, and Matthias Schubert. Probabilistic resource route queries with reappearance. In Gustavo Alonso, Floris Geerts, Lucian Popa, Pablo Barceló, Jens Teubner, Martín Ugarte, Jan Van den Bussche, and Jan Paredaens, editors, *Proceedings of the 18th International Conference on Extending Database Technology, EDBT 2015, Brussels, Belgium, March 23-27, 2015*, pages 445–456. OpenProceedings.org, 2015.

[20] Gregor Jossé, Matthias Schubert, and Hans-Peter Kriegel. Probabilistic parking queries using aging functions. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 452–455. ACM, 2013.

[21] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. In *Advances in Neural Information Processing Systems*, pages 6348–6358, 2017.

[22] Joon-Seok Kim, Dieter Pfoser, and Andreas Züfle. Distance-aware competitive spatiotemporal searching using spatiotemporal resource matrix factorization (GIS cup). In Farnoush Banaei Kashani, Goce Trajcevski, Ralf Hartmut Güting, Lars Kulik, and Shawn D. Newsam, editors, *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*, pages 624–627. ACM, 2019.

[23] Sven Koenig and Maxim Likhachev. D* lite. *AAAI/IAAI*, 15, 2002.

[24] Wouter Kool, Herke van Hoof, and Max Welling. Attention, learn to solve routing problems! In *International Conference on Learning Representations*, 2019.

[25] Jan Karel Lenstra and AHG Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11(2):221–227, 1981.

[26] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, pages 983–994, 2019.

[27] Minne Li, Zhiwei Qin, Yan Jiao, Yaodong Yang, Jun Wang, Chenxi Wang, Guobin Wu, and Jieping Ye. Efficient ridesharing order dispatching with mean field multi-agent reinforcement learning. In *The World Wide Web Conference, WWW 2019, San Francisco, CA, USA, May 13-17, 2019*, 2019.

[28] Kaixiang Lin, Renyu Zhao, Zhe Xu, and Jiayu Zhou. Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1774–1783, 2018.

[29] Iain Little, Sylvie Thiebaux, et al. Probabilistic planning vs. replanning. In *ICAPS Workshop on IPC: Past, Present and Future*, 2007.

[30] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang. Traffic flow prediction with big data: A deep learning approach. *IEEE Transactions on Intelligent Transportation Systems*, 16(2):865–873, 2015.

[31] Shuo Ma, Ouri Wolfson, and Bo Xu. Updetector: Sensing parking/unparking activities using smartphones. In *Proceedings of the 7th ACM SIGSPATIAL international workshop on computational transportation science*, pages 76–85, 2014.

[32] H Brendan McMahan, Maxim Likhachev, and Geoffrey J Gordon. Bounded real-time dynamic programming: Rtdp with monotone upper bounds and performance guarantees. In *Proceedings of the 22nd international conference on Machine learning*, pages 569–576. ACM, 2005.

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[34] Mohammadreza Nazari, Afshin Oroojlooy, Lawrence Snyder, and Martin Takác. Reinforcement learning for solving the vehicle routing problem. In *Advances in Neural Information Processing Systems*, pages 9839–9849, 2018.

[35] Layla Pournajaf, Daniel A. Garcia-Ulloa, Li Xiong, and Vaidy Sunderam. Participant privacy in mobile crowd sensing task management: A survey of methods and challenges. *SIGMOD Rec.*, 44(4):23–34, May 2016.

[36] Ulrike Ritzinger, Jakob Puchinger, and Richard F. Hartl. A survey on dynamic and stochastic vehicle routing problems. *International Journal of Production Research*, 54(1):215–231, 2016.

[37] Stuart Russell and Peter Norvig. Ai a modern approach. *Learning*, 2(3):4, 2005.

[38] Thobias Sach, Korinna Jörling, Bastian Lotz, Martin Jakob, Henrik Schult, and Diego Bietenholz. Klimaschutz in Zahlen – Fakten, Trends und Impulse deutscher Klimapolitik – Ausgabe 2020. Online: `https://www.bmu.de/fileadmin/Daten_BMU/Pools/Broschueren/klimaschutz_zahlen_2020_broschuere_bf.pdf`, 2020.

[39] Peter Sanders and Dominik Schultes. Highway hierarchies hasten exact shortest path queries. In Gerth Stølting Brodal and Stefano Leonardi, editors, *Algorithms - ESA 2005, 13th Annual European Symposium, Palma de Mallorca, Spain, October 3-6, 2005, Proceedings*, pages 568–579, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[40] Divya Saxena and Jiannong Cao. D-GAN: deep generative adversarial nets for spatio-temporal prediction. *CoRR*, abs/1907.08556, 2019.

[41] Sebastian Schmoll, Sabrina Friedl, and Matthias Schubert. Scaling the dynamic resource routing problem. In Walid G. Aref, Michela Bertolotto, Panagiotis Bouros, Christian S. Jensen, Ahmed Mahmood, Kjetil Nørvåg, Dimitris Sacharidis, and Mohamed Sarwat, editors, *Proceedings of the 16th International Symposium on Spatial and Temporal Databases, SSTD 2019, Vienna, Austria, August 19-21, 2019*, pages 80–89. ACM, 2019.

[42] Sebastian Schmoll, Sabrina Friedl, and Matthias Schubert. Orientation: Option-based reinforcementlearning for spatial navigation. Published in appendix, October 2020.

[43] Sebastian Schmoll and Matthias Schubert. Dynamic resource routing using real-time dynamic programming. In Jérôme Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 4822–4828. ijcai.org, 2018.

[44] Sebastian Schmoll and Matthias Schubert. Dynamic resource routing using real-time information. In Michael H. Böhlen, Reinhard Pichler, Norman May, Erhard Rahm, Shan-Hung Wu, and Katja Hose, editors, *Proceedings of the 21st International Conference on Extending Database Technology, EDBT 2018, Vienna, Austria, March 26-29, 2018*, pages 501–504. OpenProceedings.org, 2018.

[45] Sebastian Schmoll and Matthias Schubert. Vision paper: reinforcement learning in smart spatio-temporal environments. In Farnoush Banaei Kashani, Erik G. Hoel, Ralf Hartmut Güting, Roberto Tamassia, and Li Xiong, editors, *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, pages 81–84. ACM, 2018.

[46] Sebastian Schmoll and Matthias Schubert. Semi-markov reinforcement learning for stochastic resource collection. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3349–3355. ijcai.org, 2020.

[47] Wei Shao, Flora D Salim, Tao Gu, Ngoc-Thanh Dinh, and Jeffrey Chan. Travelling officer problem: Managing car parking violations efficiently using sensor data. *IEEE Internet of Things Journal*, 2017.

[48] Donald C. Shoup. Cruising for parking. *Transport Policy*, 13(6):479–486, 2006.

[49] Marius M Solomon and Jacques Desrosiers. Survey paper—time window constrained routing and scheduling problems. *Transportation science*, 22(1):1–13, 1988.

[50] Xiaocheng Tang, Zhiwei (Tony) Qin, Fan Zhang, Zhaodong Wang, Zhe Xu, Yintai Ma, Hongtu Zhu, and Jieping Ye. A deep value-network based approach for multi-driver order dispatching. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, page 1780–1790, 2019.

[51] Paolo Toth and Daniele Vigo. *The vehicle routing problem.* SIAM, 2002.

[52] Christopher John Cornish Hellaby Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge, UK, May 1989.

[53] Sung Wook Yoon, Alan Fern, Robert Givan, and Subbarao Kambhampati. Probabilistic planning via determinization in hindsight. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pages 1010–1016, 2008.

[54] K.J Åström. Optimal control of markov processes with incomplete state information. *Journal of Mathematical Analysis and Applications*, 10(1):174 – 205, 1965.

Acknowledgements

Without the people around supporting me, I would not have been able to contribute this work to the research community. So I want to thank everyone who brought me all the way on this site. My in-depth gratitude goes to Matthias Schubert. Thank you so much for supporting my ideas and contributing valuable comments to our discussions. Thank you for letting me benefit from your comprehensive, yet in-deep knowledge in this area. I want to thank Tomas Seidl. Not only, he found my position, but also he accompanied me on along my way as a PhD student. With your regular meetings (Doktorandenrunde) and your knowledge about interpersonal communication, among many other things, my personality is enhanced. I am very thankful to Herke van Hoof and Goce Trajcevski, for your interest in my work and for agreeing in acting as a reviewer of my thesis. My sincere appreciation is due to the chair members, who never hesitated in helping me when I had questions. Thank you Felix Borutta, Niklas Strauß, Florian Richter, Janina Sontheim, Niklas Strauß, Evgeniy Faerman, Julian Busch, Max Berrendorf, Daniyal Kazempour, Sandra Obermeier, Anna Beer, Sahand Sharifzadeh and David Winkel. Especially Sabrina Friedl, thank you for all the discussions and fruitful work we had together. Susanne Grienberger, thank you for always answering my organizational questions.

Finally, I am very grateful to all the support I received from my family over the years. In particular, I want to thank my wife and son for forbearance when I had to do extra hours before deadlines. Many thanks to my parents, as they always have an open ear for me.