Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Computer Science

Route and Charging Planning for Electric Vehicles

Doctoral thesis

Ing. Marek Cuchý

Ph.D. programme: Electrical Engineering and Information Technology
Branch of study: Information Science and Computer Engineering

Supervisor: Doc. Ing. Jiří Vokřínek, Ph.D.
Supervisor-specialist: Doc. Ing. Michal Jakob, Ph.D.

Prague, February 2024

Thesis Supervisor:
Doc. Ing. Jiří Vokřínek, Ph.D.
Department of Computer Science
Faculty of Electrical Engineering
Czech Technical University in Prague
Technická 2
160 00 Prague 6
Czech Republic

Copyright © February 2024 Ing. Marek Cuchý

Abstract

The advent of electric vehicles (EVs), brings new challenges for travel planning. In contrast to the users of combustion engine vehicles, who enjoy long driving ranges and ubiquitous and almost instant re-fuelling possibilities, EV users need to consider battery status, EV range, and dynamically changing recharging options when planning their EV trips.

In this thesis, we focus on three main challenges of EV travel planning that were motivated by real-world use cases investigated within a large interdisciplinary research project: multiple objectives, multiple destinations and incomplete information. First, we study the problem with multiple time-constrained destinations. Planning EV travel plans within the scope of a single trip has its limitations and does not allow, for example, to optimize the travel plan for the entire day. This extension of optimization scope beyond a single trip brings greater flexibility and the ability to optimize EV travel plans more effectively. Second, we study the EV travel planning problem while considering the possibility of incomplete information about charging infrastructure. The existing route EV travel planning algorithms rely on complete information availability at the time of the search. This assumption is not always valid in practice since the information may be considered business-sensitive and the providers may limit access to it. Finally, we study the EV travel planning problem optimizing multiple objectives since existing algorithms for EV travel planning rely on single-objective optimization, which limits their ability to consider EV users' multiple and often conflicting objectives, such as travel time and cost.

We provide a unified formal definition encompassing all three problem variants together with a base algorithmic approach that can be easily specialized for each of the problem variants. We evaluate all the approaches on realistic instances based on real-world data and provide insights into the properties of these emerging EV travel planning problems. We also provide an evaluation of the performance of the proposed algorithms and the impact of individual speed-up techniques on the overall performance and solution quality. We show, besides others, that the multi-destination approach provides a significant improvement in all measured properties of the EV travel plans. We can also achieve practically usable planning times within seconds with only a minor loss of solution quality, despite the very high computational complexity of the multi-objective EV travel planning problem.

The results presented in this thesis provide a solid foundation for future research in the area of EV travel planning and can be used as a starting point for the development of new, more advanced algorithms and tools for EV travel planning that can solve all the above challenges together.

Keywords: Multi-objective, Route planning, Charging Planning, Electric vehicles, Multiple destinations, Incomplete information

Abstrakt

Vzestup využití elektrických vozidel (EV) přináší nové výzvy pro plánování cest. Na rozdíl od uživatelů vozidel se spalovacím motorem, kteří mohou využívat dlouhé dojezdy a všudypřítomné a téměř okamžité možnosti doplňování paliva, musí uživatelé elektromobilů při plánování svých cest zvážit stav baterie, dojezd elektromobilu a dynamicky se měnící možnosti dobíjení.

V této práci se zaměřujeme na tři hlavní výzvy plánování cest pro EV, které byly motivovány reálnými případy použití zkoumanými v rámci velkého interdisciplinárního výzkumného projektu: více optimalizačních kritérií, více destinací a neúplná informace. Nejprve se zabýváme problémem s více destinacemi, která mají daná časová omezení, během kterých se musí navštívit. Plánování cest pro EV, které zohledňuje pouze jedné cestu má svá omezení a neumožňuje např. optimalizovat plán na celý den. Rozšíření optimalizace nad rámec jedné cesty přináší větší flexibilitu a schopnost efektivněji optimalizovat cesty pro EV. Dále se zabýváme problémem plánování cest pro EV, kde zároveň zohledňujeme možnost neúplných informací o nabíjecí infrastruktuře. Stávající algoritmy plánování pro EV se spoléhají na úplnou dostupnost informací v době vyhledávání. Tento předpoklad však v praxi nemusí vždy platit, protože informace mohou být považovány za citlivé pro podnikání a provozovatelé nabíjecích stanic k nim mohou omezit přístup. Zabýváme se také vícekriteriálním problémem plánování cest pro EV, protože stávající algoritmy spoléhají na optimalizaci pouze jednoho kritéria, což omezuje jejich schopnost zohledňovat více často protichůdných požadavků uživatelů EV, jako jsou například délka trvání cesty a co nejmenší náklady na cestování.

V této práci jsme navrhli jednotnou formální definici zahrnující všechny tři varianty problému spolu se základem algoritmu, který lze snadno specializovat na každou z variant problému. Všechny algoritmy vyhodnocujeme na realistických instancích problému založených na reálných datech a poskytujeme vhled do vlastností těchto nových variant problémů plánování cest pro EV. Poskytujeme také vyhodnocení výkonu navržených algoritmů a vlivu jednotlivých zrychlujících technik na celkový výkon a kvalitu řešení. Ukazujeme mimo jiné, že přístup s více destinacemi poskytuje výrazné zlepšení všech měřených vlastností cestovních plánů pro EV. Námi navržený algoritmus také umí dosáhnout prakticky využitelných plánovacích časů v rámci několika sekund s pouze malou ztrátou kvality řešení, a to i přes velmi vysokou výpočetní složitost vícekriteriálního problému plánování cest pro EV.

Klíčová slova: vícekriterální, plánování cest, plánování nabíjení, elektrická vozidla, více destinací, neúplná informace

Acknowledgements

I would like to thank my supervisor Michal Jakob who guided me through most of my research and provided me with valuable feedback and advice. I would also like to thank Jiří Vokřínek who supervised me during the final and most busy stages of my studies. Furthermore, I would like to thank my colleagues, namely Jan Mrkos and David Fiedler, for their support. Finally, I would like to thank all my friends, my family and most importantly my wife who supported me during my studies and especially during the final month of writing this thesis.

Contents

Chapter 1

Introduction

Electric vehicle (EV) travel planning is a complex problem that has been gaining attention in recent years due to the increasing popularity of electric vehicles. In contrast to the users of combustion engine vehicles, who enjoy long driving ranges and ubiquitous and almost instant re-fuelling possibilities, EV users need to consider EV battery status, EV range, and dynamically changing recharging options when planning travel with their EV. The route the EV user decides to take and the selected timing, speed, and location of charging can significantly impact the travel time and cost of the trip. Furthermore, the EV user's choices also impact the EV battery's health and even the percentage of renewable electricity used for charging.

Therefore, finding an optimal *EV travel plan*[1] that reflects all these issues is a complex task that can be hardly solved without AI-powered EV travel planning tools. Ideally, such tools should automatically make route and charging stop suggestions that optimize the EV user's various, often conflicting objectives and goals (e.g., fast and cheap travel plans will often not go together).

EV travel planning comprises many challenges that need to be addressed. In this thesis, our primary focus and contribution is the multi-objective EV travel planning optimizing travel time and cost. However, before we considered multiple objectives during our research, we first studied a problem that extends a trip between an origin and a single destination to a plan for a whole day via multiple temporally constrained destinations[2]. We also studied the difficulties posed by the fact that the information about charging stations and their services does not have to be always completely available. Although we did not study the problem of incomplete information and multiple destinations in such depth as the multiple-objective problem, we believe that the results of our research in

[1] We use the term *EV travel planning* for the complex task that involves optimizing both the routes the EV should take as well as the charging sessions it should make.

[2] This research was also motivated by the participation in a large international research project Electrific, which explored how AI planning and resource allocation can be employed to improve the efficiency of EV charging on the system level. `https://cordis.europa.eu/project/id/713864`

these areas are also valuable and can be used as a basis for further research. Moreover, the knowledge and experience gained from solving these problems were very important for the development of the multi-objective EV travel planning algorithm and the achieved results.

1.1 EV Travel Planning Challenges

In this section, we describe the three studied challenges in more detail.

Multiple objectives An EV user planning a trip from Passau to Hamburg in Germany (around 800 km journey) can check online whether the charging stations (CSs) along their route are working and their price, and use satellite navigation to drive between the CSs. Experienced users will use a dedicated application[3] that will find the fastest route with charging stops. However, the user cannot easily determine how much the trip will cost or whether there are cheaper options. The same trip may cost half if the driver selects a different charging station with a slight detour. Therefore, a trip planned by a route planner that optimizes only travel time might be rather costly.

Weighing trade-offs between price and speed is not a new problem in general transportation (e.g., bus vs. airplane), but it is not something most drivers of combustion engine vehicles had to consider in the past. However, using an EV for long-distance trips changes the situation significantly. This shift in planning behavior could be achieved seamlessly if EV drivers had a tool that plans their route (including charging stops) and presents them with different options for travel time and cost. The user could then select an option best fitting their needs.

Despite the significant progress in *EV travel planning* in recent years, the existing algorithms cannot yet fully support the above-described use case. The main limitation of existing approaches is that they mostly rely on *single-objective* optimization [e.g., Baum et al., 2019a] and are therefore technically limited to always considering only a single objective when finding optimal EV travel plan. Well-established approaches to multi-objective optimization, such as meta-heuristics, can find the Pareto-set only on very small city-sized road networks. Consequently, these approaches are *not* suitable for practically usable EV travel planning [e.g., Ben Abbes et al., 2022]. Very recent work of Schoenberg and Dressler [2023] achieved good planning times while considering multiple simpler objectives (not including cost) on country-scale road networks. Although their approach is similar to ours, it prohibits planning with a realistic number of charging stations. The simple solution of using scalarization of objectives (i.e., weighted sums), leads to only a

[3]Such as the built-in navigation tool in the Tesla EVs

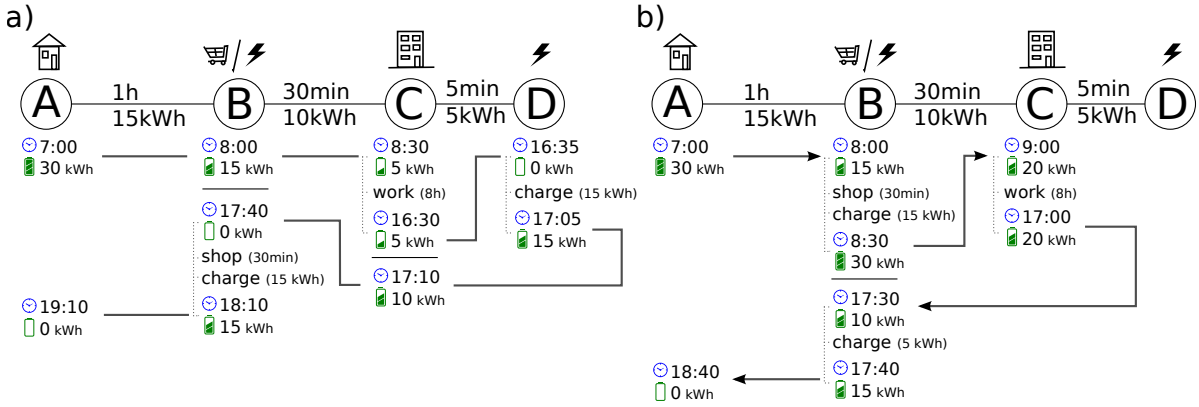


Figure 1.1: **Example scenario:** A - home location, C - work location, B, D - chargers, B - shop. Destinations are 8h work, 30min shop. Maximal state of charge (SoC) - 30 kWh, charging to full takes 60 minutes. **a) Naive approach:** The user first decides the order of the destinations (work, shop). Charging postponed until necessary. **b) Multi-destination approach:** Optimize the order of destinations and charging (shop,work).

single travel plan with a fixed objective trade-off and therefore does not provide the user with a variety of distinct options to choose from. Even if we use multiple sets of weights to generate multiple plans, the objective trade-offs are the same for each trip while in reality, the trade-offs vary significantly especially for contradictory objectives such as the time and cost of charging.

Multiple destinations The scope of EV travel planning does not have to be limited only to a single trip. Consider a busy day when the EV user needs to, for example, pick up kids from school after work, do some shopping and visit a friend in the evening. These chains of trips to multiple destinations can sum up to a significant distance that would require charging during the day. Even if the EV range is sufficient for one day of traveling, a lot of EV users do not have the option to charge the EV during the night at home and consequently start the daily commute without a fully charged battery and the necessity to charge during the day. By considering multiple trips together, EV users could organize their travel in such a way that the necessary EV charging, implied by their mobility needs, could happen at times and locations where charging can be collocated with their destination stops, the use of EVs would be more convenient. Even more can be achieved by not defining the order of the destinations ahead but defining only their temporal constraints – time windows when and how long the user wants to be at each destination.

Take for example the scenario in Figure 1.1. The user starts and ends in the home location A, may charge the EV at B and D and shop at B. The user's goal is to spend 8 hours at the workplace and to shop for 30 minutes. The initial (and maximal) capacity

of the EV battery is 30 kWh and charging to full takes 60 minutes. In the naive plan shown in Figure 1.1a), the user first decides the order of destinations, that is, first goes to work and then does the shopping. Also, the charging is postponed until necessary. By this approach, the user first goes to location C (the EV has enough charge to do that) and works for 8 hours. Next, the user wants to go home and make a stop for shopping, but the charge of the EV is not high enough to do so and thus the user must first go to a nearby charging station at location C. Then the user can get to location B and do the shopping, while also charging the EV. Finally, the user gets home, with the overhead of time caused by charging of 45 minutes (we do not count the charging time while shopping).

By optimizing only with the temporal constraints of the problem, the user can obtain the optimized plan shown in Figure 1.1b), where the shopping is scheduled before work. In that case, the user first arrives at B, does the shopping while recharging the battery to full and continues to work. On the way back, the EV does not have enough charge for the whole trip and thus a short (10 min.) charging stop is scheduled. Overall, the user arrives 30 minutes earlier than in the naive case and spends only 10 minutes on charging overhead. Notice also, that the total energy consumed from the charging stations is 10 kWh less which might also save money. This simple problem is easy to optimize, but the problem gets too complicated for a human when the number of destinations increases and the temporal constraints are more complicated. For some types of destinations, such as shops, the EV user might not care which specific subsidiary of the shop chain they visit. Therefore, their plan can be also improved by visiting the specific subsidiary that has an available and/or faster charging station nearby.

Incomplete information Besides the challenges mentioned above, there is also the challenge related to the availability of the information required to compute route recommendations for such complex and dynamic transport systems. Existing planning algorithms assume that complete information about available transport services is stored in the search graph prior to the search. This assumption is no longer realistic in transport systems including competing providers who are not willing to disclose complete information about some attributes of their services – such as charging station availability or price – as this could benefit their competitors. In such cases, the providers are often only willing to respond to narrowly scoped queries. Consequently, route and charging planning algorithms need to be adapted to be able to work with incomplete information and with limitations on how much additional information can be retrieved from service providers.

1.2 Goals of the Thesis

The main goal of this thesis is to develop a set of algorithms tackling the challenges of the EV travel planning problem posed above that enables the development of advanced driver assistance systems that make the usage of EVs more convenient and efficient.

Above, we mentioned three main challenges we aim to address in this thesis: multiple objectives, multiple destinations, and incomplete information. Although the ultimate goal of the research is to develop an approach that solves all of these challenges together, it is beyond the scope of this thesis. Therefore, we decided to focus on each challenge individually while keeping the practical relevancy.

Goal 1: Multi-destination EV travel planning problem. The extended optimization scope stemming from the consideration of multiple destinations provides the EV planning tool with greater flexibility and the ability to optimize EV travel plans more effectively. The goal is to develop an algorithm that can use this flexibility to optimize the EV travel plan in a way that is not possible with the single-destination approach.

Goal 2: EV travel planning problem with incomplete information. The common assumption that all information is available assumption is not always valid in practice since the information may be considered business-sensitive and the providers may limit access to it. In such cases, the algorithms need to work with as little additional information as possible. The goal is to develop an algorithm that minimizes the number of queries to the service provider. The information incompleteness in our case concerns the waiting times at charging stations since they reveal information about service availability and demand in space and time. The algorithm should solve the problem on real-world country-scale data in a scenario where it is possible to book a charging station ahead of time since this is probably the most complex scenario. It is also important to evaluate the impact of the problem properties on the number of queries to the service provider.

Goal 3: Multi-objective EV travel planning problem. EV users are commonly concerned with the duration and the cost of the travel plan which often go against each other and the EV user needs to consider trade-offs between them. Therefore, the final and main goal of this thesis is to develop a genuinely multi-objective algorithm that solves the multi-objective EV travel planning problem and provides the EV user Pareto-set of EV travel plans. The algorithm should achieve practically usable planning times on real-world country-scale problem instances. Moreover, an additional goal is to perform an extensive evaluation of the algorithm and the problem parameters to provide valuable insights into the problem for future research.

1.3 Thesis Outline

In Chapter 2, we review the related work and position our work in the context of the existing research. Chapter 3 provides the necessary background information about the EV travel planning problem and the used algorithms. In Chapter 4, we formally define the general EV travel planning problem that encompasses all the novel concerns, i.e., multiple destinations, multiple objectives, and incomplete information. In Chapter 5, we describe a general algorithm that is used as a basis for solving all of the problem sub-variants.

In the following three chapters, we focus on the individual problem sub-variants in the order in which we studied them. Chapter 6 focuses on the multi-destination EV travel planning problem, the algorithm we designed for solving it, and the evaluation of the multi-destination approach and the algorithm. In Chapter 7, we describe the EV travel planning problem with incomplete information and the algorithm solving it. We also provide the evaluation of the impact of problem properties on the number of queries. Chapter 8 is dedicated to the multi-objective EV travel planning problem. We describe the specifics of the problem and the algorithm that solves it including many speed-up techniques. We also provide an extensive evaluation of the algorithm and the problem parameters.

Finally, in Chapter 9, we summarize the contributions of the thesis and provide suggestions for future work.

Chapter 2

Related Work

As mentioned in the introduction, we focus on three main concerns of the EV travel planning problem. The first is the multi-objective optimization of travel time and cost, the second is the multiple destinations with time windows, and the third is the need to deal with limited access to information necessary for the planning.

In general, the first concern can be viewed as a *multi-objective* route planning problem on large road networks *constrained* by the short-range EV battery and the resulting need to stop at *charging* stations. The second concern brings the traveling salesman problem component into the problem while the third concern requires careful consideration of what information is worth gathering. Below, we give an overview of the relevant approaches, starting from the general algorithms for general graphs. These form the basis of most route planning solutions. We also describe route planning algorithms historically developed for combustion engine vehicles and how they are limited for use in EV planning. Then, we focus on many versions of single-objective EV travel planning problems and the difficulties they pose. Many of which our proposed problem variants tackle. We also describe other multi-objective approaches to planning within the context of EVs and how our work extends the state of the art. Finally, we discuss the traveling salesman aspect of the problem and the incomplete information.

2.1 Shortest Path on Generic Graphs

Most vehicle route planning algorithms have their origins in algorithms for solving the standard single-objective shortest path problem (problem in P), which can be traced as far back as to Dijkstra's algorithm [Dijkstra, 1959].

Constrained shortest path problems [Aneja and Nair, 1978] extend the classical shortest path problem to a situation when the path needs to fulfill additional constraints. The ability to handle constraints is essential for EV route planning because of the need to keep

the EV state of charge within valid bounds. In contrast with the standard, unconstrained single-objective shortest path problem, constrained shortest path problems are generally NP-hard [Garey and Johnson, 1979].

The multi-objective version of the unconstrained shortest path problem has been first addressed by the label-setting algorithm [Martins, 1984]. The multi-objective label-setting algorithms need to maintain a whole set of Pareto-optimal labels in their open queue—instead of a single label per node used by single-objective algorithms. The number of paths in the Pareto-set can be exponential in the size of the graph, making the problem inherently very hard (NP-hard but not even in NP) [Müller-Hannemann and Weihe, 2006]. Classical informed-search A* algorithm [Hart et al., 1968] has been extended to the multi-objective, unconstrained setting in the MOA* algorithm [Stewart and White III, 1991] and later improved by NAMOA* algorithm [Mandow et al., 2005]. All these multi-objective algorithms can be straightforwardly modified to solve the constrained version.

Based on the above, the EV travel planning problem is hard to solve. However, road graphs used in routing have special properties that make the shortest path search more efficient than in generic graphs.

2.2 Route Planning on Road Transport Networks

Moving from generic graphs to graphs representing road networks, researchers focused on developing a wide range of search speed-ups exploiting the specific hierarchical, quasi-planar structure of road transport networks [Eppstein and Goodrich, 2008]. Decades of research resulted in speed-ups as high as 10^6 compared to the baseline Dijkstra's algorithm, enabling sub-millisecond route planning times on continental-sized road networks [Bast et al., 2016].

Unfortunately, except for a few exceptions [e.g., Hrnčíř et al., 2016; Zhu, 2022], which study multi-objective bicycle routing, the vast majority of research on road network route planning deals with the single-objective, unconstrained formulation of the route planning problem and, as such, it is not directly applicable for EV travel planning.

Contraction Hierarchies [Geisberger et al., 2008] are a common speed-up pre-processing technique used in road network route planning. It iteratively contracts/removes nodes from the graph and creates shortcuts that maintain the shortest path distances between the remaining nodes. It was used in the context of EV route planning [Baum et al., 2019b] and Baum et al. [2019a] also adapted it to pre-calculate non-dominated shortcuts storing both travel duration and energy consumption. Planning with charging stops is also possible with contraction hierarchies.

Another potentially applicable method is presented by Delling and Wagner [2009].

The work proposes a multi-objective adaptation of SHARC algorithm [Bauer and Delling, 2009] that combines highway hierarchies [Sanders and Schultes, 2006] and arc-flags [Möhring et al., 2007] techniques. However, the arc-flag technique is unsuitable for planning with charging stops. This is because the arc-flag technique divides the graph into many partitions with a small number of boundary edges. It assigns a set of flags to each boundary edge that says if the edge lies on the shortest path to other partitions (to at least one node in the given partition). Such information is hard to utilize since optimal EV plans detour from the shortest path to recharge the battery at charging stations.

2.3 Planning for Electric Vehicles

EV-specific planning has become an active area of research in the past decade. Researchers have studied many variants of EV planning problems, differing in, e.g., whether energy consumption is treated as an optimization objective or considered, in conjunction with the state of charge (SoC) of the EV battery, only as a constraint.

One of the first approaches focused on finding the most energy-efficient routes without considering charging stops. Although this is the simplest variant of the EV planning problem, it still prohibits the use of the label-setting Dijkstra's algorithm because of the possible presence of negative edges due to energy recuperation. To circumvent this problem, Artmeier et al. [2010] proposed a solution utilizing the Bellman-Ford algorithm [Bellman, 1958] and the label-correcting version of Dijkstra's algorithm. Sachenbacher et al. [2011] propose a different approach to address the presence of negative edges; their approach first removes the negative edges by the *potential shifting* technique [Johnson, 1977] and then runs an A* search on the newly created graph. Schönfelder et al. [2014] extend the problem of finding the most energy-efficient route by searching not only for a single solution for a given initial SoC but rather for the *consumption profile* piecewise linear function that computes the optimal consumption and route for any possible initial SoC.

However, the mid-trip charging stops are essential for planning trips exceeding the range of the EV battery and/or optimizing the EV charging over multiple days. The first algorithm capable of planning charging in EV trips was proposed by Storandt and Funke [2012]. The limitation of this work is that the EV is always charged to the full battery capacity. This assumption significantly simplifies the problem.

The problem with planning EV charging stops is that there is virtually an infinite number of target SoCs to consider for each visited charging station. The A*-based algorithm finding the most energy-efficient EV travel plan proposed by Baum et al. [2019b] addresses this issue with charging by exploiting *consumption profiles* between charging stops, which allows to significantly reduce the number of generated planning states while

still maintaining optimality. The core idea of the algorithm is that the generation of labels representing various charging options at a charging station is postponed until the next charging station. This allows the algorithm to leverage the additional information about the consumption between the two charging stations to significantly reduce the originally infinite number of potentially generated labels.

The author also extended this approach in [Baum et al., 2019a] and designed an algorithm that solves the problem with SoC only as a constraint and travel time as the objective and finding the shortest feasible EV travel plan [also studied by Storandt, 2012]. Baum et al. [2019a] also employs realistic charging models that were first extensively studied by Zündorf [2014]. Charging models are important because the time required to charge an EV usually differs significantly based on the SoC. For example, it is usually much slower to charge the EV battery from 80% to 100% than it is from 20% to 40%. In these works, the authors model this behavior with a charging function that could be, for example, piecewise linear.

The real-world energy consumption of an EV is highly dependent on many difficult-to-estimate variables, such as the driving style of the driver and the condition of the vehicle. Therefore, Rajan et al. [2021] and Ünal et al. [2022] propose approaches that take into account this uncertainty.

2.4 Multi-Objective Planning for Electric Vehicles

Common approaches for solving multi-objective problems, such as genetic algorithms [Ben Abbes et al., 2022] or particle swarm optimization [Siddiqi et al., 2011], were applied to EV travel planning. Although the authors consider the cost of charging in these works, the methods were evaluated only on very small road networks with only hundreds of nodes. Realistic road graphs required in EV route planning have millions of nodes. As such, these techniques do not currently scale to realistic problem instances.

Genetic algorithms and other meta-heuristics are also used for more complex multi-objective problems where the route and charging in EV planning are solved only as sub-problems, such as distribution of charging stations [Tran et al., 2021] or design of the electric public transit network [Liu et al., 2020]. However, individual EV planning in these problems is often oversimplified, so the results cannot be used in practice by EV drivers. For example, they commonly simplify the routing part, where they consider only one pre-calculated route between charging stations and/or other points of interest (e.g., bus stops). Furthermore, the graphs used in these works contain orders of magnitude less nodes than needed for practical applications.

Schoenberg and Dressler [2023] proposed an algorithm based on multi-objective A*

and a set of speed-ups that finds a Pareto-optimal set of plans while minimizing time and energy consumption. Although the proposed algorithm can find the Pareto-set in milliseconds, the solved problem is much simpler. In our proposed problem, we need to consider three different attributes to be non-dominated - two objectives (time and cost) and one constraint (energy consumption). The problem proposed by Schoenberg and Dressler [2023] requires only two attributes to be non-dominated - two objectives (time and energy consumption), where the latter is also a constraint. The dimension of the non-dominated attributes has the greatest impact on the complexity of a multi-objective problem. Moreover, one of the proposed speed-ups pre-calculate the routes between all pairs of charging stations, which is possible if the number of charging stations is small. The authors considered approx. 1000 charging stations in their evaluation resulting in the data with the size of 36GB. Since the size requirements are quadratic, we can assume that the 12 thousand charging stations, we use in our multi-objective experiments (Section 8.4.1), would require approx. 244× more space, making this speed-up unusable in our problem. We made a similar observation[1] when we evaluated our first approach to road graph pre-processing while solving the multi-destination version of the problem (Section 6.4).

2.5 Traveling Salesman Problem

The multi-destination nature of the discussed problem is well incorporated in the Traveling Salesman Problem (TSP). The Traveling Salesman Problem is a classic NP-hard problem. In our case, the more relevant variant is the Steiner TSP [Cornuéjols et al., 1985] where only some of the nodes of a graph are required to be visited by the solution walk. Moreover, the edges and nodes may repeat. Our problem subsumes a combination of three extensions to the classic (or Steiner) TSP.

The first is the Generalized TSP [Rice and Tsotras, 2013], where each city to be visited is represented as a subset of the graph nodes and it is sufficient to visit one node from each subset. The second variant is the TSP with Time Windows where the cities have to be visited in a given time window. In our case, the lower bound constraint is soft, which is related to the TSP with Deadlines where there is no lower bound on the visit time and for which no constant approximation ratio can ever be achieved [Bockenhauer et al., 2007]. The main difference from our problem is that the destination visits have durations of their own which have to be spent at the destination. The last related variant of the TSP is the Resource-Constrained TSP [Pekny and Miller, 1990] where a resource is consumed when traversing an edge and the solution path cannot consume more than a given resource maximum. Our problem differs in that the resource in question (energy)

[1] Before the approach of Schoenberg and Dressler [2023] was published.

can be also replenished.

The most relevant related problem is the Electric TSP with Time Windows (E-TSPTW) [Roberti and Wen, 2016] which considers both the temporal and energy constraints but still exhibits a number of simplifications of the real problem. Similarly to the TSPTW, a destination with time window $[t^{\mathsf{min}}, t^{\mathsf{max}}]$ can be visited before t^{min} and wait. In our problem, the whole duration of the destination visit must be spent at the destination within the time window. The discharging and charging models in the E-TSPTW are greatly simplified. In the E-TSPTW, all charging stations have the same charging rate and both the discharged energy and charged energy depend linearly on the traveled distance or charging time respectively which allows for the use of linear program formulations. Our problem formulation is more realistic in that we place almost no assumptions (besides laws of physics) on the energy consumption (e.g., it can depend on the elevation profile and can even be negative for recuperation on downhill edges) and on the charging function defining how long a charging session will take. Besides the common dependence on the amount of charged energy and charging power, it allows modeling of more realistic charging behavior that slows down as the battery gets full. The more general Electric VRP with Recharging Stations and Time Windows solved by Schneider et al. [2014] share the same properties as E-TSPTW regarding our problem.

2.6 Incomplete Information

Another important aspect is the possibility of incomplete information. Information incompleteness can be modeled by general methods such as POMDP [Oliehoek and Amato, 2016] or by more specific *conformant* planning and *contingent* planning discussed by Bonet and Geffner [2000]. Both of them deal with incomplete information but the former does not allow observations that update the information, in this case, called *sensing*, while the latter does. Sensing is present also in the Canadian Traveler Problem [Nikolova and Karger, 2008] where the cost of edges is given by probability distributions and the actual cost is known only when arriving at the endpoint of an edge. Although sensing allows gathering some additional information, it does so only during plan execution; in contrast, in our case, the information can be gathered already during planning.

Even though it is possible to get the information, it can be time-consuming to obtain all the information. The queries can be only narrowly targeted, which leads to a great number of them and performing too many queries can be time-consuming either because of the communication overhead or because there is a limit on the maximum rate of queries. The incompleteness of the information in our problem can be modeled as time-consuming

edge cost computation[2].

The problem of time-consuming edge cost computation is most commonly solved in the context of robot motion planning where the calculation of possible collisions with obstacles is computationally expensive. An extensively used approach is the lazy evaluation (Lazy Weighted A* [Cohen et al., 2015], LazySP [Dellin and Srinivasa, 2016]). The lazy algorithms mostly differ only in the way the edge for evaluation is selected as it is shown in [Dellin and Srinivasa, 2016] where algorithms are compared and formulated as LazySP with different edge selectors. The authors also propose several new selectors. Minimizing edge evaluations is also solved by Partial Expansion A* [Yoshizumi et al., 2000] which is suited for problems with large branching factors. Narayanan and Likhachev [2017] propose an algorithm working in two phases. In the first phase, the algorithm efficiently finds all relevant paths, and in the second phase, the edges are evaluated in the order defined by a policy optimal for anytime interruption. The algorithm above solves problems where the edge costs are known but the existence of the edges is unknown (the existence is defined by edge existence probability). However, in our problem, also the cost is unknown besides edge existence.

A completely different approach is used by Phillips et al. [2014] which deals with the expensive evaluation by parallelization of the A* algorithm. This approach could be interesting but in the case of EV travel planning with incomplete information, the great cost is not caused by computational complexity but by communication overhead and/or a limited number of evaluations.

2.7 Summary

Despite the proliferation of work addressing different aspects of EV travel planning, none of the existing approaches support the simultaneous optimization of multiple objectives while considering the battery constraint and generating a whole Pareto-set of EV travel plans combining routes and recharging stops on country-sized road networks. Consequently, existing approaches make it difficult to properly address the trade-offs EV users can have between travel planning objectives and, in particular, properly incorporating increasingly important pricing considerations into planning EV trips.

Multiple destinations bring the traveling salesman problem component into EV travel planning. Although there are close variants of the TSP [Roberti and Wen, 2016; Schneider et al., 2014], none of them are directly applicable to our problem since they make assumptions and simplifications about energy consumption and charging behavior which

[2]In our problem, the incomplete information is not related to an edge but to charging stations. However, the problem definition could be redefined to a very complex search graph where the incomplete information would be an edge cost.

makes their formulation unsuitable for our problem.

Most of the approaches to problems considering incomplete information that is expensive to evaluate (possible to retrieve/compute in a time-consuming manner) use some form of lazy evaluation on graphs with a single unconstrained objective and a relatively small amount of expensive-to-evaluate information. However, in the context of EV travel planning, the value of the expensive-to-evaluate information is time-dependent and also the resource (battery) is constrained; such a variant of the incomplete information EV travel planning problem has not been previously explored.

Chapter 3

Preliminaries

In this chapter, we describe the basic notation, concepts and basic algorithms used in this thesis. We start with the basic graph terminology and the shortest path problem including its multi-objective variant. We also describe well-known algorithms for solving these problems and a relaxation technique that is used to speed up all the algorithms proposed in the thesis. Finally, we describe the concepts related to energy consumption and battery constraints.

3.1 Notation

In the thesis, we adhere, with some exceptions, to the following notation principles. Scalars and singular elements are defined by lower case letters (for example, a, b, c). Sets or collections are denoted by corresponding capital letters (A, B, C). Functions are denoted by Greek letters (α, β, γ). More general tuples are written in calligraphic font ($\mathcal{A}, \mathcal{B}, \mathcal{C}$). We also tried to use corresponding letters for related concepts. For example, a time value is denoted as lower case t and a function returning time is Greek τ.

3.2 Graph Terminology

A *directed graph* is a pair $G = (V, E)$ where V is a finite set of nodes and $E \subseteq \{(u, v) | u, v \in V\}$ is a finite set of edges. An edge $e \in E$ is an ordered pair of vertices $e = (u, v)$. An *undirected graph* $G = (V, E)$ contains edges consisting of unordered pairs of vertices $E \subseteq \{\{u, v\} | u, v \in V\}$ instead of ordered pairs. A graph is *simple* if it contains at most one edge between any pair of nodes. A *multigraph* is a graph that contains multiple edges between any pair of nodes, i.e., E is a multiset of edges. For simplicity, further on we assume, if not mentioned otherwise, that all graphs are simple and directed. A graph is *weighted* if each edge $e \in E$ has assigned cost $\omega : E \to \mathbb{R}$, for example, length or traversal

time. Multiple costs can be assigned to each edge. We define *outgoing* edges of node $v \in V$ as a set of all edges starting at v. Analogously, we define *incoming* edges of node $v \in V$ as a set of all edges ending at v. We also define a backward graph of a directed graph $G = (V, E)$ as $\overleftarrow{G} = (V, \overleftarrow{E})$ where $\overleftarrow{E} = \{(v, u) | (u, v) \in E\}$ is the set of the edges with reversed direction.

A *path* $P = (v_1, v_2, \ldots, v_k)$ is a sequence of nodes where $(v_i, v_{i+1}) \in E$ for each $i \in \{1, \ldots, k-1\}$. Let $G = (V, E, \omega)$ be a weighted graph and $P = (v_1, v_2, \ldots, v_k)$ be a path in graph G. Cost of path P denoted as $\omega(P) = \sum_{i=1}^{k-1} \omega(v_i, v_{i+1})$ is a sum of the edge costs between the path nodes.

3.3 The Shortest Path Problem

A *shortest path* P between two nodes $u, v \in V$ is a path starting at node $u = v_1$ and ending at node $v = v_k$ such that the path cost $\omega(P)$ is minimal among all the possible path between the nodes in graph G. The cost of the shortest path between u, v according to the cost ω is denoted as $\omega(u, v)$.

Dijkstra's algorithm [Dijkstra, 1959] solving so-called *single-source shortest path problem* (shortest path from a single node to all others) is outlined in Algorithm 1.

Algorithm 1: Pseudocode of Dijkstra's algorithm calculating shortest path costs from an origin to all other nodes.

Input: weighted graph $G = (V, E, \omega)$
origin $o \in V$
Output: Shortest path costs

```
1  function Dijkstra
2      Q: priority queue
3      W: array of costs for each node
4      V^cl: set of closed nodes
5      W[v] = ∞, ∀v ∈ V
6      W[o] = 0
7      Q ← {⟨o, 0⟩}
8      V^cl ← ∅
9      while Q ≠ ∅ do
10         u ← extractMin(Q)
11         V^cl ← V^cl ∪ {u}
12         forall (u, v) ∈ E do
13             if W[u] + ω(u, v) < W[v] and v ∉ V^cl then
14                 W[v] ← W[u] + ω(u, v)
15                 insertOrUpdate(Q, v, W[v])
16     return W
```

The algorithm uses three basic data structures:

- Priority queue Q ordering the explored nodes by the associated cost, usually implemented as a heap.
- Array W storing costs of the shortest paths from the origin to all nodes.
- Set of already settled/closed nodes V^{cl}.

In each iteration, the algorithm extracts/removes the node with the smallest cost u from the priority queue and adds it to the set of closed nodes. For each outgoing edge of the node, it calculates the cost to the neighboring node v and checks if it is smaller than the cost already stored in $W[v]$. If the cost is smaller, the stored cost is updated and the node is added to the queue. If the node is already in the queue, the cost associated with it in the queue is only updated.

The algorithm has a so-called *label-setting* property meaning that once a node u is removed/extracted from the priority queue its cost can never change and is equal to the shortest path cost $W[u] = \omega(s, u)$. We say that the node is *settled* or *closed*. However, the algorithm is correct only if the edge costs are non-negative ($\omega : E \to \mathbb{R}_0^+$). If only a path/cost to a single destination is required the algorithm can be terminated after extracting the destination node from the queue.

If we remove the closed node set check simply by removing the line 11 from the algorithm, it can be used also with negative edge costs. Although this modification changes the polynomial asymptotic complexity to exponential, it outperforms the $O(|V||E|)$ complexity of Bellman-Ford algorithm [Bellman, 1958] in practice on graphs with a small percentage of negative edge costs (which is the case we use it in this thesis)[Artmeier et al., 2010]. We call this modification of Dijkstra's algorithm *label-correcting*.

The algorithm returns only the costs, but it can be easily adapted to store also the previous nodes needed for the shortest paths reconstruction.

3.4 The Multi-Objective Shortest Path Problem

As a generalization of the shortest path problem can be seen the *multi-objective shortest path problem*. Instead of a single cost $\omega(e)$, multiple costs $\omega_1(e), \ldots, \omega_k(e)$ is assigned to each edge and analogously a path P has multiple costs $\omega_1(P), \ldots, \omega_k(P)$. The goal is to minimize the path costs. Since there is more than one optimization objective, a total ordering does not usually exist. For example, path A takes 2 hours and costs 10€, while path B takes 90 minutes and costs 20€. We cannot simply order the paths by their cost and duration. Path A is cheaper, but path B is faster. There is a trade-off between the two objectives.

However, a partial ordering exists according to *weak Pareto dominance*:

Definition 1. Let $s = \langle w_1, \ldots, w_k \rangle, s' = \langle w'_1, \ldots, w'_k \rangle$ be two tuples representing costs of two paths. We say that s *weakly dominates* s' (denoted as $s \preceq s'$) iff $w_i \leq w'_i, \forall i \in \{1, \ldots, k\}$.

Further, we refer to the *weak dominance* only as the *dominance* for simplicity. In cases where neither dominates the other ($s \npreceq s'$ and $s' \npreceq s$), we say that they are *non-dominated*.

We also refer to the cost tuples, or any extension of them, as *states*. We also introduce the dominance between a state and a set of states.

Definition 2. Let s be a state and S be a set of states. We say that S dominates s (denoted as $S \preceq s$) iff

$$\exists s' \in S : s' \preceq s \tag{3.1}$$

Algorithm 2: Pseudocode of multi-objective variant of Dijkstra's algorithm

Input: weighted graph $G = (V, E, \omega_1, \ldots, \omega_k)$
origin $o \in V$
Output: Pareto-optimal set of shortest paths costs for each node

function Multi-Objective-Dijkstra
  S_v^{op}: set of opened states for each graph node $v \in V$
  S_v^{cl}: set of visited/closed states for each graph node $v \in V$
  $S^{\mathsf{op}} = \bigcup_{v \in V} S_v^{\mathsf{op}}$: set of all opened states
  $S_v^{\mathsf{op}} \leftarrow \emptyset, \forall v \in V$
  $S_v^{\mathsf{cl}} \leftarrow \emptyset, \forall v \in V$
  $S_o^{\mathsf{op}} \leftarrow \{\langle 0, \ldots, 0 \rangle\}$
  $S_o^{\mathsf{cl}} \leftarrow \{\langle 0, \ldots, 0 \rangle\}$
  while $S^{\mathsf{op}} \neq \emptyset$ **do**
    $(u, \langle w_1, \ldots, w_k \rangle) \leftarrow$ `extractMin(`S^{op}`)`
    $S_u^{\mathsf{cl}} \leftarrow S_u^{\mathsf{cl}} \cup \{\langle w_1, \ldots, w_k \rangle\}$
    forall $(u, v) \in E$ **do**
      $s \leftarrow \langle w_1 + \omega_1(u, v), \ldots, w_k + \omega_k(u, v) \rangle$
      if $(S_v^{\mathsf{op}} \cup S_v^{\mathsf{cl}}) \npreceq s$ **then**
        $S_v^{\mathsf{op}} \leftarrow S_v^{\mathsf{op}} \cup \{s\}$
  return $S_v^{\mathsf{cl}}, \forall v \in V$

Extension of single-source Dijkstra's algorithm to a multi-objective setting [Hansen, 1980; Martins, 1984] is outlined in Algorithm 2. The algorithm uses three basic types of data structures:

- Pareto-set of closed/visited states S_v^{cl} for each graph node $v \in V$ that contains all states that were already visited by the algorithm.

- Pareto-set of opened states S_v^{op} for each graph node $v \in V$ that holds the states that were generated but not yet visited by the algorithm.
- Set of all opened states $S^{\mathsf{op}} = \bigcup_{v \in V} S_v^{\mathsf{op}}$, that can also be viewed as a priority queue for the states to be visited.

In each iteration, the algorithm extracts a non-dominated state (commonly lexicographical minimum) from the priority queue S^{op}. It adds the state to the corresponding closed set S_u^{cl}. For each outgoing edge of the node, it calculates the costs to the neighboring node v and checks if it is not dominated by any of the opened or closed states. If it is not dominated, it is added to the opened set S_v^{op} which means it is also added to the priority queue S^{op}. Once the priority queue is empty, the algorithm stops and the closed Pareto-sets contain the Pareto-optimal sets of states representing paths to the corresponding nodes.

The algorithm can be modified for the single-destination setting by checking if the extracted node and state is the destination and adding it to a Pareto-set of solutions. The set of solutions can be also used to further prune the generated states. If a newly generated state is dominated by any of the already found solutions we know that its costs cannot improve on the remaining path to the destination (assuming non-negative costs).

3.5 Informed Search and Heuristics

We can modify the single-destination versions of the algorithms described above to use additional information to guide the search, enhancing Dijkstra-based algorithms to A* algorithms [Hart et al., 1968; Stewart and White III, 1991].

The algorithm estimates the remaining cost to the destination (e.g., direct distance) and orders the priority queue accordingly. If $v \in V$ is a node, and w is a cost associated with the node, and $h : V \to \mathbb{R}_0^+$ is the heuristic estimate function, the priority queue is sorted by $w + h(v)$ instead of w.

We use the multi-objective version of the A* algorithm as a basis for all the proposed algorithms in this thesis. Therefore, we provide a detailed description of the multi-objective version of the A* algorithm specialized for the EV travel planning problem defined in Chapter 4 in its dedicated Chapter 5.

To ensure the correctness of the A* algorithm, the heuristics have to be *admissible* which means that the estimated cost cannot be greater than the real shortest path cost.

Definition 3. Let s be a state in a *planning problem state space*. We say that heuristic h is *admissible* iff $h(s) \leq h^*(s)$ for all possible states, where $h^*(s)$ is the remaining optimal cost to the destination s.

Another important property of heuristics is *consistency* (or *monotonicity*).

Definition 4. Let s, s' be two consecutive states in a *planning problem state space*. We say that heuristic h for a single objective is *consistent* iff $h(s) \leq \omega(s, s') + h(s')$, where $\omega(s, s')$ is the objective cost for the transition between the two states.

Every consistent heuristic is also admissible.

3.6 Dominance Relaxation

The dominance relation is a fundamental concept in multi-objective optimization. It allows us to at least partially order the states in the multi-objective space. However, the dominance relation is not a total ordering and instead of a single best state, we have a whole set of Pareto-optimal states that the algorithms must explore. Since the number of explored states is the main factor influencing the algorithm performance, multi-objective optimization is much more challenging than single-objective optimization.

Therefore, one of the possible algorithm speed-ups is to relax the dominance relation such that more states are considered dominated and therefore pruned. This can be done by a so-called ϵ-dominance relaxation [Laumanns et al., 2002; Batista et al., 2011] that modifies the dominance relation from Definition 1 as follows:

Definition 5. Let $s = \langle w_1, \ldots, w_k \rangle, s' = \langle w'_1, \ldots, w'_k \rangle$ be two cost tuples. We say that s *ϵ-dominates* s' for $\epsilon_i \in [0, 1], \forall i \in \{1, \ldots, k\}$ (denoted as $s \preceq_\epsilon s'$) iff $\epsilon_i \cdot w_i \leq w'_i, \forall i \in \{1, \ldots, k\}$.

The number of pruned states and therefore the speed-up is dependent on the ϵ coefficients. The relaxation can also prune some of the Pareto-optimal solutions; and therefore, does not preserve optimality. Consequently, the ϵ coefficients must be chosen carefully according to the desired trade-off between the algorithm performance and the quality of the results.

3.7 Energy Consumption and SoC Profiles

The energy consumption and battery constraints of electric vehicles can be modeled very straightforwardly. For EV travel planning purposes, we can model the energy consumption $\beta(e) \in \mathbb{R}$ as a constant[1] assigned to each graph edge $e \in E$ and check if the SoC does not

[1]The exact value can be provided by any consumption model based on the properties of the road segment the edge represents such as elevation profile, length or speed.

drop below 0 during the planning. The consumption can be calculated, for example, by the following linear model similar to the one used by Eisner et al. [2011]:

$$\beta(e) = \alpha^l l(e) + \alpha^+ elev^+(e) - \alpha^- elev^-(e) \tag{3.2}$$

where l is the edge length, and $elev^+$ and $elev^-$ are the elevation gain and elevation drop accumulated during the traversal of the edge e in meters and where $\alpha^l, \alpha^+, \alpha^-$ are the model coefficients.

Note that the consumption can be negative due to recuperation which leads to the increase of the battery state of charge unless the battery is already fully charged. Such a case can also be easily handled by the planning algorithm.

If the initial SoC is known the planning is straightforward. The algorithm just prunes states with SoC below 0 and caps the SoC to battery capacity.

Unfortunately, the final SoC and the energy consumption of a route are dependent on the initial SoC. For example, a route going through a mountain pass that starts with a steep ascent and ends with a long descent will require a much higher initial SoC than the simple consumption calculated as the difference between the initial and final SoC of the route. During the ascent, the EV will consume a lot of energy while the descent recharges the battery thanks to recuperation. With the initial SoC equal to the simple route energy consumption, the EV would deplete the battery during the ascent.

A similar case is when the route starts with a descent recharging the battery. The final SoC of the route would be the same if the EV was fully charged at the start of the route or if it was slightly discharged because the EV could not recuperate the energy if the battery was already fully charged.

Therefore, we need to calculate the *SoC profile* first studied by Schönfelder et al. [2014] and later used by **?**Baum et al. [2019b] that provides optimal consumption for each possible initial SoC. The following definitions are based on the formulation by Baum [2018]. The SoC profile for a path p can be defined as a function $\rho_p : [0, b_{\mathsf{max}}] \to [0, b_{\mathsf{max}}] \cup -\infty$ that returns the final SoC for each possible initial SoC or $-\infty$ if the path is infeasible. The profile can be defined by just four values: minimal required initial SoC $\mathsf{in}_p \in [0, b_{\mathsf{max}}]$, maximal final SoC $\mathsf{out}_p \in [0, b_{\mathsf{max}}]$, the energy cost $\mathsf{cost}_p \in [-b_{\mathsf{max}}, b_{\mathsf{max}}]$ and the battery capacity $b_{\mathsf{max}} \in \mathbb{R}^+$ which is the same for all profiles. Based on these values, the SoC profile can be defined as follows:

$$\rho_p(b) = \begin{cases} -\infty & \text{if } b < \mathsf{in}_p, \\ \mathsf{out}_p & \text{if } b - \mathsf{cost}_p > \mathsf{out}_p, \\ b - \mathsf{cost}_p & \text{otherwise} \end{cases}$$

For a single edge $e \in E$ with energy consumption $\mathsf{cost}_e \in [-b_{\mathsf{max}}, b_{\mathsf{max}}]$, the two unknown SoC profile attributes can be calculated as follows: $\mathsf{in}_e = \max(0, \mathsf{cost}_e)$ and $\mathsf{out}_e = \min(b_{\mathsf{max}}, b_{\mathsf{max}} - \mathsf{cost}_e)$.

Since we need it for the calculation of sequences of edges/routes, we need to concatenate the SoC profiles. Let ρ_p, ρ_r be two SoC profiles for paths p, r then their concatenation $\rho_{p \circ r} = \rho_p \circ \rho_r$ is defined as follows:

$$\begin{aligned} \mathsf{in}_{p\circ r} &= \max(\mathsf{in}_p, \mathsf{cost}_p + \mathsf{in}_r) \\ \mathsf{out}_{p\circ r} &= \min(\mathsf{out}_p - \mathsf{cost}_r, \mathsf{out}_r) \\ \mathsf{cost}_{p\circ r} &= \max(\mathsf{cost}_p + \mathsf{cost}_r, \mathsf{in}_p - \mathsf{out}_r) \end{aligned}$$

SoC profiles can also be easily checked for dominance.

Definition 6. Let ρ_p, ρ_q be two SoC profiles. We say that ρ_p *dominates* ρ_q (denoted as $\rho_p \preceq \rho_q$) iff

$$\begin{aligned} \mathsf{in}_p &\leq \mathsf{in}_r \\ \mathsf{out}_p &\geq \mathsf{out}_r \\ \mathsf{cost}_p &\leq \mathsf{cost}_r \end{aligned}$$

Since we have defined dominance relation and concatenation of SoC profiles, we can use a simple modification of the multi-objective Dijkstra's algorithm described above to find Pareto-optimal SoC profiles.

Chapter 4

EV Travel Planning Problem Definition

In this chapter, we formally define the general EV travel planning problem that encompasses all the variants of the problem we consider in this thesis. We model the EV travel planning problem as a multi-objective constrained shortest path problem with SoC constraints and charging stops with two optimization objectives: time and cost. We define the EV travel planning problem as a tuple $\mathcal{P} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R} \rangle$ where $\mathcal{W}$ is the global static *EV travel planning environment* (Section 4.1), $\mathcal{M}$ is the *EV model* (Section 4.2), and $\mathcal{R}$ is the *EV travel planning request* (Section 4.3) that is specific for each EV user and their needs. The solution to an EV travel planning problem is the Pareto-set of *EV travel plans* Π (Sections 4.4 and 4.5). Schema of the problem is depicted in Figure 4.1.

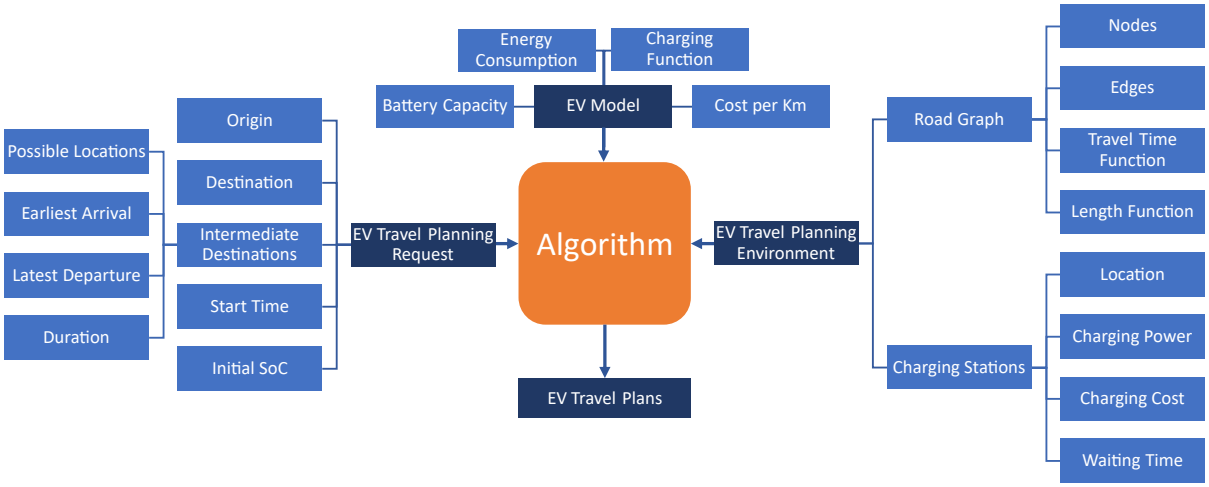


Figure 4.1: Schema of the multi-objective multi-destination EV travel planning problem.

4.1 EV Travel Planning Environment

The EV travel planning environment (termed *planning environment* further on) represents the road network and charging stations, i.e., the components of the travel planning problem that are independent of the specific details of individual planning requests.

The planning environment is a tuple $\mathcal{W} = \langle G, Q \rangle$, where $G = \langle V, E, \tau, l \rangle$ is a weighted oriented graph representing the underlying road network, with V being the set of graph nodes representing intersections and E the set of graph edges representing road segments. Each edge $e \in E$ has a defined traversal duration $\tau(e) \in \mathbb{R}^+$ and a length $l(e) \in \mathbb{R}^+$. Although the algorithm does not require it, we assume for the sake of simplicity that the graph G is without loops and parallel edges.

The set of charging stations Q defines the locations where EVs can be charged. Each charging station $q \in Q$ is defined as a tuple $q = \langle v_q, P_q, \tau_q, \gamma_q \rangle$, where $v_q \in V$ is the node where the charging station is located, $P_q \in \mathbb{R}^+$ is the maximum power the charging station provides (*charging rate*), $\tau_q : \mathbb{R}_0^+ \times \mathbb{R}^+ \to \mathbb{R}_0^+$ is the *waiting time* how long the EV user will have to wait at the given time $t \in \mathbb{R}_0^+$ for a charging session of duration $\delta^{\mathsf{ch}} \in \mathbb{R}^+$ before the charging station is available, and $\gamma_q : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}_0^+$ is the *charging cost function* that defines how much any charging session at the station q costs based on the duration $\delta^{\mathsf{ch}} \in \mathbb{R}^+$ of the session and the amount of energy $j \in \mathbb{R}^+$ charged during the session. We also define a set of all nodes where the charging stations are located as $V_Q = \{v_q | q \in Q\}$.

The *charging cost function* can formalize various types of charging policies, including all of those popular today, such as fixed price per charging session, price per minute of charging, price per kWh of charged energy, or their combination. Since each charging station q may have defined its own cost function γ_q, our proposed formalism also allows modeling the *location-of-use* pricing, where the price of charging depends on the location of the charging station. In fact, the algorithmic approach we propose to solve the EV travel planning problem can work with even more complex types of arbitrary non-negative pricing policies, in particular with the *time-of-use* pricing, where cost functions γ_q depend on the time of charging. Although totally compatible with our approach, we do not further consider *time-dependent* pricing functions, primarily for the sake of simpler presentation and also because there are not yet enough real-world data that could be used for the evaluation of EV travel planning with time-dependent charging prices.

The *waiting time* function needs to consider also the duration of charging since it models the charging station reservation system, where the EV users can reserve a time slot at the charging station in advance. Since there can be gaps between reserved time slots, long charging requests do not have to fit into the gaps. The function can also model the queues and the time-dependent availability of the charging station, where the charging

station is not available at certain times of the day.

4.2 EV Model

The electric vehicle model $\mathcal{M} = \langle b_{\mathsf{max}}, \beta, \phi, \psi, B \rangle$ consists of the *maximum battery capacity* $b_{\mathsf{max}} \in \mathbb{R}^+$ of the EV, *cost per km of driving* $\psi \in \mathbb{R}_0^+$, the *target charging levels* $B \subset (0, b_{\mathsf{max}}]$ defining charging levels to be considered by the algorithm, and two functions defining how the EV consumes the energy stored in its battery and how the battery is recharged.

The *energy consumption function* $\beta : E \times [0, b_{\mathsf{max}}] \to [0, b_{\mathsf{max}}] \cup \{-\infty\}$ defines the SoC after traversing edge $e \in E$ while depending on starting SoC. The energy consumption function can take into account various properties of the edge, such as the length or elevation profile. In fact, the consumption can be an arbitrary function (black box) that follows the laws of physics (no negative cycles). The consumption can be negative due to recuperation. $-\infty$ means that the starting SoC is too low to traverse the edge.

The *charging function* $\phi : [0, b_{\mathsf{max}}] \times (0, b_{\mathsf{max}}] \times \mathbb{R}^+ \to \mathbb{R}^+$ defines the time needed to complete a charging session specified by the starting SoC $b_{\mathsf{start}} \in [0, b_{\mathsf{max}}]$, the final SoC $b_{\mathsf{end}} \in (0, b_{\mathsf{max}}]$ and the maximum available power $P_{\mathsf{max}} \in \mathbb{R}^+$. An example of a simplified representation of a piecewise linear charging function is given in Figure 4.2. As already mentioned, the time required to charge EVs usually differs significantly based on the starting SoC. For example, it is usually much slower to charge the battery from 80% to 100% of battery capacity than from 20% to 40% of battery capacity. The charging function models this behavior.

The *cost per km of driving* $\psi \in \mathbb{R}_0^+$ defines EV wear and tear costs per driven distance. We introduced it to explicitly account for wear-and-tear costs as this provides a more accurate model of the real-world optimization problem faced by EV drivers. Moreover, it helps to avoid unreasonably long detours to free charging stations. Without the driving cost, the most cost-efficient route would be a route through a series of free charging stations even though it would require absurd detours.

The *target charging levels* $B \subset (0, b_{\mathsf{max}}]$ define the final SoC levels to be considered for charging. For example, charging to 80%, 90%, 100% of battery capacity. We use the discretization of the target charging levels to significantly reduce the number of newly generated states. Instead of considering a theoretically infinite number of charging amounts, we only consider a small set of predefined target charging levels when expanding the charge action in the search. We consider this discretization a reasonable simplification because in practical scenarios the difference between charging 10% or 11% of battery capacity has a very small impact on the resulting plans.

That said, the discretization is still a limitation of our approach. Baum et al. [2019b]

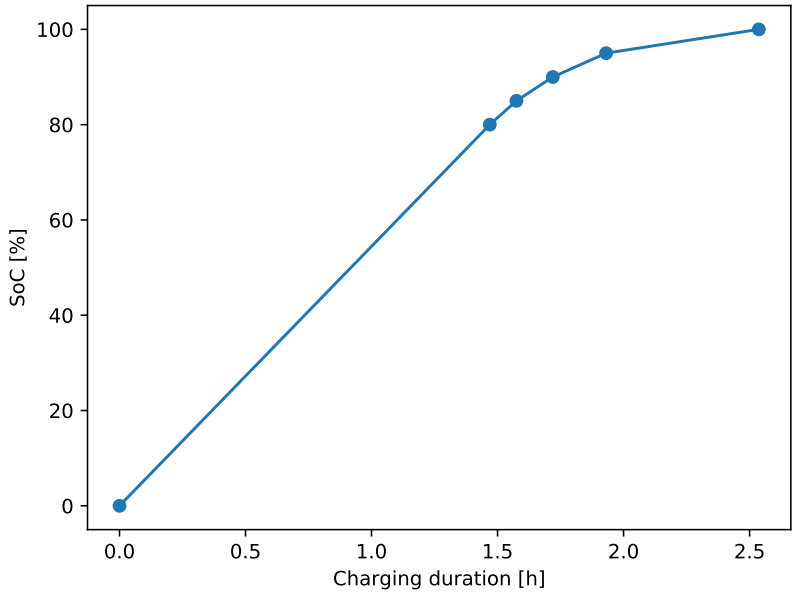


Figure 4.2: Piecewise linear charging function for an EV with 40kWh battery on 20kW charging station. It is a simplified representation depicting the charging duration starting from the empty battery. The duration of charging starting from a different SoC can be calculated as a difference between the duration of charging from an empty battery to the target SoC and the duration of charging from an empty battery to the starting SoC.

propose an approach that avoids the discretization of target charging levels and which is able to find the exact amount of energy to be charged without having to generate a very high and in fact infinite number of possible charging levels. The approach is based on replacing the SoC state attribute with a *SoC profile* function (described in 3.7) that enables postponing the state generation to the next charging station and generating the states based on the additional information about the consumption on the route between the charging stations. With this additional information, the approach derives the set of charging options in an optimal way. However, the approach is only proposed for the single-objective version of the problem (no charging cost), and it is not in the presented form applicable to the multi-objective version. Although we study single-objective variants of EV travel planning in this thesis, the main contribution is the multi-objective variant. The approach described above is rather complicated and we concluded that it would be more beneficial to focus on other improvements of the single-objective algorithms. Therefore, we did not incorporate the solution proposed by Baum et al. [2019b] to the algorithms proposed in this thesis.

The set of target charging levels can be configured arbitrarily, but it should take into account the shape of the charging function ϕ.

Let $b \in [0, b_{\mathsf{max}})$ be a state of charge, we define a subset of target charging levels

feasible for charging at b as $B_{>b} = \{b' \in B | b' > b\}$. We also define the amounts of charged energy feasible for charging starting at b as $J^B_{>b} = \{b' - b | b' \in B_{>b}\}$

4.3 EV Travel Planning Request

The EV travel planning request defines the user's specific request for EV travel planning. The request is defined as a tuple $\mathcal{R} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, D, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$, where $v_{\mathsf{init}} \in V$ is the origin, $v_{\mathsf{goal}} \in V$ is the final destination, $t_{\mathsf{init}} \in \mathbb{R}^+_0$ is the start time, $b_{\mathsf{init}} \in [0, b_{\mathsf{max}}]$ is the initial SoC, and where D is the set of intermediate destinations. An intermediate destination $d \in D$ is a tuple $d = \langle V_d, t_d^{\mathsf{min}}, t_d^{\mathsf{max}}, \delta_d \rangle$ where:

- $V_d \subset V$ is the set of graph nodes at which the destination stop is possible,
- $t_d^{\mathsf{min}} \in \mathbb{R}^+_0$ is the earliest possible arrival at the destination,
- $t_d^{\mathsf{max}} \in \mathbb{R}^+_0$ is the latest possible departure from the destination,
- $\delta_d \in \mathbb{R}^+_0$ is the duration the EV user stays at the destination.

The intermediate destinations are optional and can be used to model the user's daily schedule. An intermediate destination does not have to be defined by only one location but can be defined by a set of locations (e.g., a set of supermarket branches of a given brand). The intermediate destinations are specified by the time window with the earliest arrival, the latest departure and the duration of the stay such that: $t_d^{\mathsf{min}} \leq t_d^{\mathsf{max}}$ and $\delta_d \leq t_d^{\mathsf{max}} - t_d^{\mathsf{min}}$.

4.4 EV Travel Plan

The *EV travel plan* for a vehicle model $\mathcal{M} = \langle b_{\mathsf{max}}, \beta, \phi, \psi, B \rangle$ in a planning environment $\mathcal{W} = \langle G, Q \rangle$ and a planning request $\mathcal{R} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, D, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ is a sequence of interleaving states and actions $\pi = (s_0, a_0, s_1, a_1, \ldots, a_{k-1}, s_k)$.

A *state* s_i fully describes the status of the EV and the value of plan objectives at the i-th step of the plan and action a_i describes the transition between the states s_i and s_{i+1}.

We define the state s as a tuple $\langle v, t, c, b, D_{\mathsf{rem}} \rangle$ where:

- $v \in V$ is an EV location graph node,
- $t \in \mathbb{R}^+_0$ is the time at which the state is reached,
- $c \in \mathbb{R}^+_0$ is the charging and driving cost spent to reach the state,

- $b \in [0, b_{\mathsf{max}}]$ is the SoC with which the state is reached (higher value means more energy in the battery),
- $D_{\mathsf{rem}} \subseteq D$ is the set of remaining destinations to be visited.

An EV travel plan consists of four types of *actions*:

- *move(e)* that moves the EV across the edge $e = (v, u) \in E$:

$$\langle v, t, c, b, D_{\mathsf{rem}} \rangle \rightarrow \langle u, t + \tau(e), c + \psi l(e), \beta(e, b), D_{\mathsf{rem}} \rangle$$

- *charge(q, j)* that charges the EV at the charging station $q \in Q$ with energy $j \in J^B_{>b}$:

$$\langle v_q, t, c, b, D_{\mathsf{rem}} \rangle \rightarrow \langle v_q, t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}, c + \gamma_q(\delta^{\mathsf{ch}}, j), b + j, D_{\mathsf{rem}} \rangle$$

  where $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session and $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ is the waiting time before charging.

- *visit-destination(d)* that visits destination $d \in D_{\mathsf{rem}}$ at node $v_d \in V_d$ such that $t + \delta_d \leq t^{\mathsf{max}}_d$:

$$\langle v_d, t, c, b, D_{\mathsf{rem}} \rangle \rightarrow \langle v_d, \max(t + \delta_d, t^{\mathsf{min}}_d + \delta_d), c, b, D_{\mathsf{rem}} \setminus \{d\} \rangle$$

- *visit-destination-and-charge(d, q, j)* that visits destination $d \in D_{\mathsf{rem}}$ and charges the vehicle at the charging station $q \in Q$ with energy $j \in J^B_{>b}$ at node $v_q = v_d \in V_d$ such that $t + \delta_d \leq t^{\mathsf{max}}_d$:

$$\langle v_d, t, c, b, D_{\mathsf{rem}} \rangle \rightarrow \langle v_d, \max(t_d, t + \delta^{\mathsf{ch}} + \tau_q(t, \delta^{\mathsf{ch}})), c + \gamma_q(\delta^{\mathsf{ch}}, j), b + j, D_{\mathsf{rem}} \setminus \{d\} \rangle$$

  where $t_d = \max(t + \delta_d, t^{\mathsf{min}}_d + \delta_d)$ is the destination visit end time, and $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session, and $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ is the waiting time before charging.

Since a state fully describes the EV status and the values of optimization objectives, the last state of a plan can also be used as a simpler representation of the plan. For example, the state $s = \langle v, 950\text{s}, 5\,€, 10\text{kWh}, \{d_1, d_3\} \rangle$, represents a plan that takes 950 seconds, costs 5 €, has two unvisited intermediate destinations and the EV ends at node v with 10kWh of energy in the battery.

In order for the EV travel plan $\pi = (s_0, a_0, s_1, a_1, \ldots, a_{k-1}, s_k)$ to be valid, the state of charge must not drop below zero or get above the maximum battery capacity b_{max}: $0 \leq b_i \leq b_{\mathsf{max}}, \forall i \in 0, \ldots, k$.

We say that an EV travel plan π with $k+1$ states is feasible for a planning request $\mathcal{R} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, D, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ if it is valid, $v_0 = v_{\mathsf{init}}, t_0 = t_{\mathsf{init}}, b_0 = b_{\mathsf{init}}$, $v_k = v_{\mathsf{goal}}$, $D_{\mathsf{rem},0} = D$ and $D_{\mathsf{rem},k} = \emptyset$. We also define the *plan time* as $t_\pi = t_k$ and the *plan cost* as $c_\pi = c_k$.

4.4.1 EV Travel Plan Objectives and Dominance

An EV travel planning algorithm solving problem $\mathcal{P} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R} \rangle$ should produce EV travel plans feasible plans for planning request $\mathcal{R}$ optimal with regard to two objectives – time and cost. More specifically, the goal of the algorithm is to minimize t_π and c_π.

Since there is more than one optimization objective, a total ordering with regard to t_π and c_π does not usually exist. However, a partial ordering exists according to dominance:

Definition 7. Let π, π' be two EV travel plans. We say that π *dominates* π' (denoted as $\pi \preceq \pi'$) iff $t_\pi \leq t_{\pi'}$ and $c_\pi \leq c_{\pi'}$.

4.5 EV Travel Planning Problem Solution

The solution to the multi-objective EV travel planning problem $\mathcal{P}$ is a set of feasible Pareto-optimal (non-dominated) EV travel plans Π. The travel plans are optimal regarding the travel time t_π and the cost c_π minimization objectives.

The EV travel plans in the resulting Pareto-set Π express the possible trade-offs between the two objectives – the travel and charging time t_π and the charging and driving cost c_π.

The formal definition we have presented in this chapter is general and encompasses all the variants of the EV travel planning problem we study in this thesis. We specify the three sub-variants and how they modify the definition in their respective chapters: the single-objective multi-destination variant in Chapter 6, the variant with incomplete information in Chapter 7, and the multi-objective single-destination variant in Chapter 8.

Chapter 5

Algorithm

In this chapter, we describe the algorithm that is a baseline for all the algorithms solving the problem sub-variants described in the following chapters. The algorithm described in this chapter is a version of the multi-objective A* algorithm, more specifically NAMOA* [Mandow et al., 2005], that is applied to the EV travel planning problem defined in the previous chapter. You can see its pseudocode in Algorithm 3.

The core of the algorithm is the same for all the problem sub-variants except for several details. The main difference is in the state expansion and the state space pruning. The algorithms also use different heuristic functions (used mostly for state ordering) and other speed-up techniques.

At first, we focus on several variants of state dominance used in the algorithm, then on the data structures the algorithm uses, and finally we focus on the description of the main steps of the algorithm.

5.1 States and Their Dominance

To describe the algorithm, we use the same definition of states $s = \langle v, t, c, b, D_{\mathsf{rem}} \rangle$ as presented in Section 4.4[1]. As mentioned above, a state can also be viewed as a simpler representation of a (partial) EV travel plan since it fully describes all essential attributes that are necessary for the planning algorithm to decide about the subsequent actions. We say that a plan is *partial* if its last location is not the final destination and not all intermediate destinations are visited.

In this section, we formally extend the concept of EV travel plan *dominance* (Definition 7) to states while maintaining full compatibility. The algorithm requires two versions of the dominance that are used in different algorithm steps. π-dominance in Definition 8 is

[1] Although the reconstruction of the final plans requires additional state attributes (for example, a reference to the preceding state and charging details), we omitted them for a clearer presentation.

a straightforward adjustment of Definition 7 to the context of states leveraging the information provided by time and cost heuristics h_t and h_c. The algorithm uses π-dominance when it checks the explored states against the already found solution plans.

Definition 8. Let $s = \langle v, t, c, b, D_{\mathsf{rem}} \rangle, s' = \langle v', t', c', b', D'_{\mathsf{rem}} \rangle$ be two states. We say that s π-dominates s' (denoted as $s \preceq_\pi s'$) iff the following conditions are satisfied:

$$\begin{aligned} t &\leq t' + h_t(s') \\ c &\leq c' + h_c(s') \end{aligned} \tag{5.1}$$

However, π-dominance does not work if both states represent partial plans (not at the destination yet). For example, a state representing a partial plan that is slower but has a higher SoC could lead to a faster plan at the final destination because it could have enough energy to reach the destination without any additional stop at a charging station. Therefore, the algorithm requires the following dominance extended by the SoC attribute, set of remaining intermediate destinations, and without the heuristic estimates to check the states representing partial plans (details of how it is used in the section below). The second dominance relation in Definition 9 is used when the algorithm checks the states representing partial plans (not at the destination yet) against each other.

Definition 9. Let $s = \langle v, t, c, b, D_{\mathsf{rem}} \rangle, s' = \langle v', t', c', b', D'_{\mathsf{rem}} \rangle$ be two states at the same node ($v = v'$). We say that s dominates s' (denoted as $s \preceq s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t &\leq t' \\ c &\leq c' \\ b &\geq b' \\ D_{\mathsf{rem}} &\subseteq D'_{\mathsf{rem}} \end{aligned} \tag{5.2}$$

The definitions above are general and could be used later in the problem sub-variant algorithms. However, for the sake of clarity, we define later also the specific versions for the respective algorithms.

5.2 Algorithm Data Structures

The algorithm uses four basic types of data structures:

- Pareto-set of visited/closed states S_v^{cl} for each graph node $v \in V$ that contains all states that were already visited and expanded by the algorithm.[2]

[2]The algorithm maintains open and closed sets for all nodes to contain only non-dominated states.

- Pareto-set of opened states S_v^{op} for each graph node $v \in V$ that holds the states that were generated but not yet visited by the algorithm.[2]
- Solution set Π with the states representing plans that visited all the destinations.
- Set of all opened states $S^{\mathsf{op}} = \bigcup_{v \in V} S_v^{\mathsf{op}}$, that can also be viewed as a priority queue for the states to be visited.

5.3 Algorithm Description

Algorithm 3: Pseudocode of the general EV travel planning algorithm.

Input: planning environment $\mathcal{W} = \langle G, Q \rangle$
planning request $\mathcal{R} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, D, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$
EV model $\mathcal{M} = \langle b_{\mathsf{max}}, \beta, \phi, \psi, B \rangle$
Output: set of Pareto-optimal travel plans Π

```
1  function Plan
2      S_v^op: set of opened states for each graph node v ∈ V
3      S_v^cl: set of visited/closed states for each graph node v ∈ V
4      S^op = ⋃_{v∈V} S_v^op: set of all opened states - queue
5      Π: set of solution states
6      S_v^op ← ∅, ∀v ∈ V
7      S_v^cl ← ∅, ∀v ∈ V
8      Π ← ∅
9      S_{v_init}^op ← {⟨v_init, t_init, 0, b_init, D⟩}
10     while S^op ≠ ∅ do
11         s_min ← extractMin(S^op)
12         if Π ⪯_π s_min then
13             continue
14         if isGoal(s_min) then
15             Π ← Π ∪ {s_min}
16         else
17             S_{v_min}^cl ← S_{v_min}^cl ∪ {s_min}
18             S ← expand(s_min)
19             S ← prune(S)
20             forall s = ⟨v, t, c, b, D_rem⟩ ∈ S do
21                 if (S_v^op ∪ S_v^cl) ⪯ s ∨ Π ⪯_π s then
22                     continue
23                 else
24                     S_v^op ← S_v^op \ {s' ∈ S_v^op | s' ⪯ s}
25                     S_v^op ← S_v^op ∪ {s}
26     return Π
```

There are 6 main steps in the algorithm that are executed in each iteration of the main loop. See pseudocode in Algorithm 3.

Minimal state extraction: At first, a minimal non-dominated state $s_{\min}$ is extracted from the set of all opened states S^{op} (`extractMin` on line 11). The correctness of the algorithm only requires that the state has to be non-dominated according to the following dominance very similar to the one from Definition 9:

Definition 10. Let $s = \langle v, t, c, b, D_{\mathsf{rem}} \rangle, s' = \langle v', t', c', b', D'_{\mathsf{rem}} \rangle$ be two states. We say that s queue-dominates s' (denoted as $s \preceq_Q s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t + h_t(s') &\leq t' + h_t(s') \\ c + h_c(s') &\leq c' + h_c(s') \\ b &\geq b' \\ D_{\mathsf{rem}} &\subseteq D'_{\mathsf{rem}} \end{aligned} \tag{5.3}$$

Since the set of opened states S^{op} can contain a whole Pareto-set of such non-dominated state candidates, we need to choose a strategy by which we select one state from the candidates. There are two general strategies: we can either use the lexicographical ordering or use a linear combination of the attributes. Both strategies guarantee the extraction of a non-dominated state. We use the lexicographical strategy because it is a prerequisite of one of the used speed-ups (described later). To order the states, we can use the estimates provided by heuristic functions h_t and h_c (which makes the algorithm A*). The states $s = \langle v, t, c, b, D_{\mathsf{rem}} \rangle$ are ordered first by their estimated time $t + h_t(s)$, then by cost $c + h_c(s)$, SoC b and then by the number of remaining destinations $|D_{\mathsf{rem}}|$ and if the states are still not ordered a random one is selected.

Solution dominance pruning: Each extracted state is first checked for whether it is not π-dominated by any of the already found solution states (line 12). If it is, it is pruned immediately and the next state is extracted from the opened set S^{op}. Although the states dominated by solution states Π are also pruned immediately after they are generated (line 21), new dominating solution states could be added while the extracted state was in the opened set.

Solution check: If the state is a solution (`isGoal` on line 14), it is added to the set of solutions Π.

State expansion: If the state is neither π-dominated nor a solution itself, the state is added to the corresponding visited/closed set S_v^{cl} (line 17) and expanded (`expand` on line 18). The state expansion is the step where the specific algorithms differ the most and are therefore described in their designated chapters.

State pruning: The states generated from the expansion are then pruned (`prune` on line 19), for example, by the SoC constraints and possibly by other specific constraints of problem sub-variants.

Addition to the opened set: Finally, in the last step of the main loop, the states that are not dominated by any of the already visited states S_v^{cl}, opened states S_v^{op}, or the solution states Π (line 21) are added to the opened set S_v^{op} (line 25) while removing the states that are dominated by the newly added state (line 24).

Termination and result: The algorithm terminates when the opened set S^{op} is empty and returns the set of found solution states Π.

The algorithm described above provides a general framework for solving the problem sub-variants. The algorithms differ mainly in the dominance relations, the state expansion and pruning steps, the termination condition and used speed-up techniques. We describe the specifics of the algorithms for each of the problem sub-variants in their respective chapters.

Chapter 6

Single-Objective Multi-Destination EV Travel Planning

In this chapter, we present a single-objective multi-destination sub-variant of the EV travel planning problem. This variant simplifies the problem by considering only a single objective, the plan duration, which significantly reduces the complexity of the problem by reducing the dimension of the Pareto-sets considered by the algorithm and also searches only for a single solution, the fastest one. Despite the simplification above, the problem is still very complex due to its traveling salesman problem nature caused by the need to visit multiple destinations.

We studied this problem variant in [Cuchý et al., 2018a] which was extended in [Cuchý et al., 2018c]. The first paper presents the multi-destination problem, the benefits of the multi-destination approach and the basic algorithm, while the second paper enhances the proposed algorithm with a pre-processing speed-up technique. Since the research of the multi-destination variant of the problem is not the main goal of the thesis, the presented approach and its evaluation have their limitations and require further research. However, we believe that the results are interesting and worth presenting.

6.1 Problem Definition

The single-objective multi-destination EV travel planning problem is defined very similarly to the general problem definition $\mathcal{P} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R} \rangle$ with the following modifications:

- The solution to the problem is a single plan π optimized only for plan duration t_π.
- The *charging cost functions* γ_q and *cost per km of driving* ψ can be ignored[1].

[1] Although we measure the cost in the evaluation, it is not a part of the problem optimization since we optimize only the plan duration. It can be seen as an additional plan property similar to, for example, traveled distance or number of charging sessions.

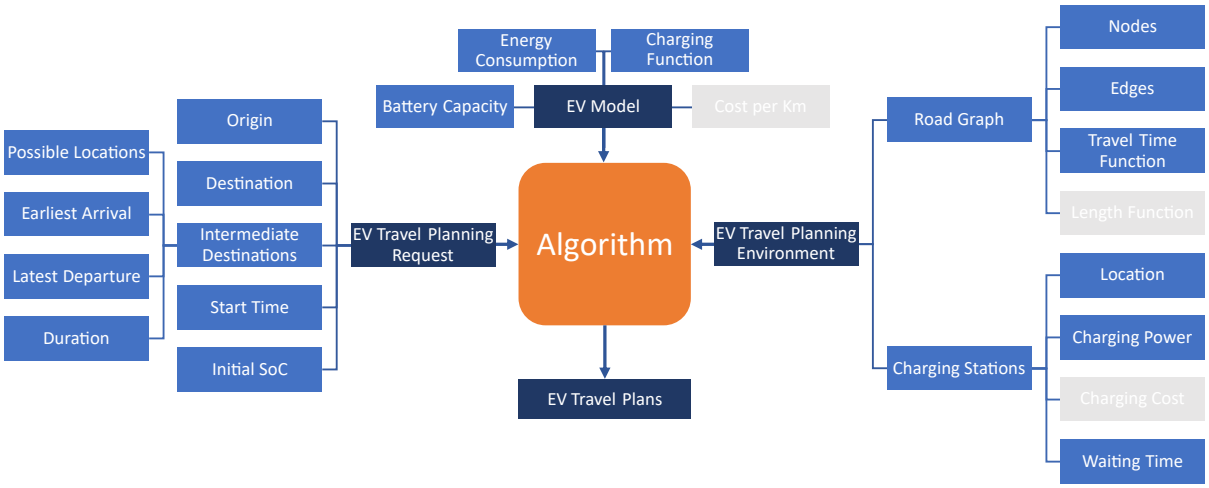


Figure 6.1: Schema of the single-objective multi-destination EV travel planning problem.

- The states and all actions of EV travel plans can ignore the cost attribute.

More specifically, the problem is defined by tuple $\mathcal{P}^{\mathsf{SOMD}} = \langle \mathcal{W}^{\mathsf{SO}}, \mathcal{M}^{\mathsf{SO}}, \mathcal{R} \rangle$ where $\mathcal{W}^{\mathsf{SO}} = \langle G, Q^{\mathsf{SO}} \rangle$ is a simplified (single-objective) environment, $\mathcal{M}^{\mathsf{SO}} = \langle b_{\mathsf{max}}, \beta, \phi \rangle$ is a simplified (single-objective) EV model, and $\mathcal{R} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, D, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ is a planning request.

The schema of the problem with greyed-out attributes that are not used in this problem variant is shown in Figure 6.1.

6.2 Single-Objective Multi-Destination Algorithm

The algorithm is based on the multi-objective algorithm presented in the previous chapter. The algorithm uses simplified states and dominance relations, specific *state expansion*, two *state pruning* techniques (Sections 6.2.1 and 6.2.2), and a specific *heuristic function* (Section 6.2.3). We also developed a *route pre-processing* speed-up technique (Section 6.2.4).

To solve the single-objective multi-destination EV travel planning problem, we use simplified states without the cost attribute: $s = \langle v, t, b, D_{\mathsf{rem}} \rangle$. This also leads to different dominance relations.

The more complex dominance relation from Definition 9 is simplified to:

Definition 11. Let $s = \langle v, t, b, D_{\mathsf{rem}} \rangle, s' = \langle v', t', b', D'_{\mathsf{rem}} \rangle$ be two states at the same node ($v = v'$). We say that s dominates s' (denoted as $s \preceq s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t &\leq t' \\ b &\geq b' \\ D_{\mathsf{rem}} &\subseteq D'_{\mathsf{rem}} \end{aligned} \tag{6.1}$$

Since the algorithm is only single-objective, the π-dominance relation (Definition 8) is not needed at all and the algorithm can terminate after finding the first solution. Therefore, we can simplify lines 12-15 in Algorithm 3 to:

---

if `isGoal(`$s_{\min}$`)` **then**
  return $s_{\min}$

The most specific parts of the algorithm are the `expand` and `prune` steps. The `expand` needs to consider all possible actions described in Section 4.4. Let $s_{\min} = \langle v, t, b, D_{\mathsf{rem}} \rangle$ be the extracted state. The state is then expanded by the following actions (according to the actions described in the problem definition Section 4.4):

(i) **move** For each outgoing edge $e = (v, u) \in E$, a new state

$$s = \langle u, t + \tau(e), \beta(e, b), D_{\mathsf{rem}} \rangle$$

is generated.

(ii) **charge** For each charging station $q = \langle v_q, P_q, \tau_q, \gamma_q \rangle$ such that $v_q = v$ and for each amount of energy $j \in J^B_{>b}$, a new state

$$s = \langle v_q, t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}, b + j, D_{\mathsf{rem}} \rangle$$

is generated, where $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session, and $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ is the waiting time before charging.

(iii) **visit-destination** For each remaining intermediate destination $d = \langle V_d, t_d^{\mathsf{min}}, t_d^{\mathsf{max}}, \delta_d \rangle \in D_{\mathsf{rem}}$ such that $v \in V_d$ and $t + \delta_d \leq t_d^{\mathsf{max}}$, a new state

$$s = \langle v, \max(t + \delta_d, t_d^{\mathsf{min}} + \delta_d), b, D_{\mathsf{rem}} \setminus \{d\} \rangle$$

is generated.

(iv) **visit-destination-and-charge** For each remaining intermediate destination $d = \langle V_d, t_d^{\mathsf{min}}, t_d^{\mathsf{max}}, \delta_d \rangle \in D_{\mathsf{rem}}$ such that $v \in V_d$ and $t + \delta_d \leq t_d^{\mathsf{max}}$, and for each charging station $q = \langle v_q, P_q, \tau_q, \gamma_q \rangle$ such that $v_q = v$ and for each amount of energy $j \in J^B_{>b}$, a new state

$$s = \langle v, \max(t_d, t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}), b + j, D_{\mathsf{rem}} \setminus \{d\} \rangle$$

is generated, where $t_d = \max(t + \delta_d, t_d^{\mathsf{min}} + \delta_d)$ is the destination visit end time, $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session, and $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ is the waiting time before charging.

We have also developed two `prune` techniques (besides the pruning below 0 SoC).

6.2.1 Temporal Consistency Forward-Checking

States from which any of the remaining destinations cannot be reached in time can be pruned. To preserve optimality we use an optimistic lower bound of a minimal time required to get to the intermediate destination. This lower bound consists not only of the estimate of the travel time but also of the estimate of the minimal time required for charging if the current state of charge is not enough. The condition which all states $s = \langle v, t, b, D_{\mathsf{rem}} \rangle$ has to satisfy is formulated as:

$$\forall d \in D_{\mathsf{rem}} \exists v_d \in V_d : t_d^{\mathsf{max}} - \delta_d \geq t + \tau_t(v, v_d) + \tau_c(v, v_d)$$

where $\tau_t(v, v_d)$ is the fastest path duration between v and v_d and $\tau_c(v, v_d) = \max(0, \beta(v, v_d) - b)/P_{\mathsf{max}}$ is the optimistic estimate of charging time with $\beta(v, v_d)$ as the energy required by the most energy-efficient path from v to v_d and $P_{\mathsf{max}} = \max_{q \in Q} P_q$ is the maximum power among all charging stations.

The fastest path duration $\tau_t(v, v_d)$ can be pre-calculated at the start of the core algorithm by multiple executions of backward Dijkstra's algorithm for all $v \in V$ and all $v_d \in V_D$. The most energy-efficient paths $\beta(v, v_d)$ can be pre-calculated too, only by a label-correcting version of Dijkstra's algorithm. This may seem like a significant overhead, but compared to the core algorithm its impact is small - especially if we parallelize the pre-calculation. However, if the number of all destination locations $|V_D|$ is too large, the overhead may be significant. In such a case, we can replace the exact calculations with an optimistic estimate based on the direct distance of the locations.

6.2.2 State of Charge Consistency Forward-Checking

This consistency checking prunes away all states from which it is impossible to get to any charging station $q \in Q$ or to the final destination v_{goal} without getting the state of charge below some minimum b_{min} (in our case we consider only $b_{\mathsf{min}} = 0$). For all states $s = \langle v, t, b, D_{\mathsf{rem}} \rangle$, the following condition has to hold

$$\exists v' \in V_Q \cup \{v_{\mathsf{goal}}\} : b - \beta(v, v') \geq b_{\mathsf{min}}$$

where $\beta(v, v_d)$ is the energy required by the most energy-efficient path from v to v_d which can be reused from the previous pruning technique.

6.2.3 Remaining Travel Time and Destination Duration Heuristic

The search is guided by a heuristic function that relaxes the SoC constraints and also the need to visit all the intermediate destinations. It uses the remaining destination via which the travel time to the final destination is the longest. It also adds the duration of all remaining destinations.

Let $s = \langle v, t, b, D_{\mathsf{rem}} \rangle$ be the currently extracted state. For each destination $d \in D_{\mathsf{rem}}$ which was not finished yet we define its estimated travel time to the final destination as

$$t_d = \min_{v_d \in V_d} (\tau(v, v_d) + \tau(v_d, v_{\mathsf{goal}})) \tag{6.2}$$

where $\tau(v, v_d)$ and $\tau(v_d, v_{\mathsf{goal}})$ are the fastest path duration between v and v_d, and v_d and v_{goal}. The fastest path durations can be reused from the pre-calculation of the temporal consistency pruning. Intuitively, t_d denotes the time spent on the fastest trip from the node v to an intermediate destination node v_d and then to the final destination v_{goal} excluding the intermediate destination stop duration.

The heuristic itself can be expressed as

$$h_t(s) = \begin{cases} \tau(v, v_{\mathsf{goal}}) & \text{if } D_{\mathsf{rem}} = \emptyset, \\ \max_{d \in D_{\mathsf{rem}}} t_d + \sum_{d \in D_{\mathsf{rem}}} \delta_d & \text{otherwise} \end{cases} \tag{6.3}$$

that is, we take the worst time of a trip via an intermediate destination to the final destination and add the sum of the durations of intermediate destinations that still need to be visited. If all intermediate destinations are visited, the heuristic returns the time of the fastest route from the current node to the final destination.

Theorem 1. *Heuristic h_t defined by Equation 6.3 is admissible.*

Proof. Let $h^*(s)$ denote the remaining duration of the optimal solution starting at state $s = \langle v, t, b, D_{\mathsf{rem}} \rangle$. To prove the admissibility of h_t, we need to show that $h_t(s) \leq h^*(s)$ for all states s.

There are two cases to consider. The first case is when there are no remaining intermediate destinations ($D_{\mathsf{rem}} = \emptyset$). In this case, the heuristic is equal to the fastest possible travel time to the final destination ignoring battery constraints, which is always less than or equal to the optimal solution that may require additional time for charging.

The second case is when there are remaining intermediate destinations ($D_{\mathsf{rem}} \neq \emptyset$). We can split the optimal solution duration into two parts: the time spent at the intermediate destinations h^*_d and the rest which includes travel time and charging time h^*_r. The time

spent at the intermediate destinations is at least the sum of the durations of all remaining intermediate destinations, therefore $h_d^*(s) \geq \sum_{d \in D_{\mathsf{rem}}} \delta_d$ which is the right part of the heuristic in Equation 6.3. The rest of the optimal solution duration $h_r^*(s)$ is at least the minimal travel time via all the remaining intermediate destinations and this cannot be faster than the minimal travel time via only one intermediate destination, therefore $h_r^*(s) \geq \max_{d \in D_{\mathsf{rem}}} t_d$ which is the left part of the heuristic in Equation 6.3.

Since both parts of the optimal solution duration are greater or equal to both parts of the heuristic, the whole optimal solution duration is always greater or equal to the heuristic ($h_t(s) \leq h^*(s)$); and therefore, admissible. □

6.2.4 Route Pre-Processing

We introduced this speed-up in [Cuchý et al., 2018c].

Most of the states generated during the search are generated by the *move* action. Therefore, we designed a speed-up based on pre-processing of routes between all charging stations and all points of interest (POIs).

We define the set of all such locations as $V^{\mathsf{POI}} \subset V$ while $V_Q \subset V^{\mathsf{POI}}$. Unfortunately, the number of routes between all these locations grows quadratically with their number and therefore the number of such locations has to be limited. Since we calculate the routes only between POIs we need to restrict the destination nodes $\forall d \in D : V_d \subset V^{\mathsf{POI}}$. This restriction appears to be reasonable since the number of general points of interest (schools, shops, offices) is limited and the destinations that are not at these preselected locations can be either mapped to the nearest POI (if the distance is not too great), or we can use first-/last-mile post-processing to get from/to the exact origin/destination in real-world applications. Since the number of POIs poses the main limitation for the practical applicability of this approach, one of the research questions is to find out how many POIs this approach can handle with reasonable memory requirements and pre-processing time.

This speed-up consists of two phases, the pre-processing phase and the query phase. During the pre-processing phase, we build a road graph with pre-calculated shortcuts E^{POI} between all pairs of POIs. The query phase then uses the same algorithm as described at the beginning of this section only with the pre-processed graph as the input.

The pre-processing phase computes and stores non-dominated shortest paths between all pairs of high-level nodes $v \in V^{\mathsf{POI}}$. The paths are non-dominated with respect to the travel time t and the required amount of energy j.

Definition 12. Let p, p' be two paths with travel time t and required amount of energy j (t' and j' respectively), then p dominates p' iff $t \leq t' \wedge j \leq j'$.

Each such path is stored as an edge e with associated traversal time $\tau(e) = t$ and

energy consumption function $\beta(e, b), b \in [0, b_{\mathsf{max}}]$ defined according to the required amount of energy j:

$$\beta(e,b) = \begin{cases} -\infty & \text{if } b - j < 0, \\ b_{\mathsf{max}} & \text{if } b - j > b_{\mathsf{max}}, \\ b - j & \text{otherwise} \end{cases}$$

To calculate the paths, we use the multi-objective Dijkstra's algorithm described in Section 3.4.

The pre-calculation of the routes as described above calculates only the total energy consumption of the route assuming that the EV always starts the route with a full battery. This assumption, unfortunately, may violate the correctness of the algorithm since for other initial SoCs the energy consumption may be different (more details in Section 3.7). However, such cases are quite rare. The negligible impact was confirmed on our evaluation instances, where the impact on plan duration is none and on energy consumption is less than 0.1%. Moreover, since the real consumption is dependent on many unpredictable variables the inaccuracy of consumption models is higher than the possible inaccuracy of the pre-calculated routes[2]. Due to very limited impact on the quality of the presented results, the issue is not further addressed.

6.2.4.1 Route Reduction by Clustering

One of the main factors in the complexity of the query algorithm is the number of all pre-calculated routes $|E^{\mathsf{POI}}|$. One possible technique to reduce the number of routes is to select only a subset of k routes between each two nodes $u, v \in V^{\mathsf{POI}}$. Here we describe how we select such a subset of size k. We propose a heuristic selection technique which unfortunately does not guarantee optimality. There is also no guarantee that the algorithm will find a solution even if it exists. There is a remote possibility of discarding all non-dominated routes that lead to the goal. Therefore, the subset should be selected with caution. The proposed subset selection proceeds as follows.

Let $E_{u,v} \subseteq E^{\mathsf{POI}}$ be the set of all non-dominated routes between u and v represented by edge $(u, v) \in E_{u,v}$. We proceed by clustering the edges in $E_{u,v}$ by each of the objectives (time, energy) and then by selecting the best route in each cluster based on the other one. For clustering, we use the well-known k-means algorithm [Hartigan and Wong, 1979].

1. Find $k/2$ clusters in $E_{u,v}$ based on travel time $\tau(e)$.

2. For each cluster select the edge e with the lowest required energy j.

[2]Except for some extremely rare cases where a POI would be on top of a very long downhill road.

The same is done for energy consumption and travel time respectively. The result is a set of k edges which are a representative sample w.r.t. one constraint and the best w.r.t. the other constraint.

6.2.4.2 Route Reduction by Dominance Relaxation

We can use the dominance relaxation technique to speed up also the pre-processing phase by using ϵ-relaxed dominance (as defined in Section 3.6) in the multi-objective Dijkstra's algorithm used to find the routes. This also results in a smaller set of all such routes.

6.3 Evaluation of Multi-Destination Approach and Basic Algorithm

Above, we described a novel multi-destination approach to EV travel planning and the algorithm that solves it. This section provides an experimental evaluation of the multi-destination approach itself by its comparison with a baseline sequential single-trip approach and an evaluation of the proposed algorithm without the pre-processing speed-up technique which we evaluate separately in Section 6.4. First, we describe the baseline approach and the set of used problem instances. Next, we evaluate the proposed multi-destination algorithm (Section 6.2) against the baseline approach in terms of the impact on the travel plan duration, consumption and charging cost (even though it is not an optimization objective). Then, we evaluate the performance impact of the proposed heuristic (Section 6.2.3) and pruning techniques (Sections 6.2.1 and 6.2.2), and evaluate the effect of dominance relaxation (Section 3.6) on the quality of the solution and planning time of the algorithm.

6.3.1 Baseline Approach

To evaluate the effect of the global multi-destination approach to solving the EV travel planning problem, we need to evaluate it against a baseline approach. The baseline is based on the same A*-based algorithm (Section 6.2) with the following modifications.

The most important modification is that, similarly to a human user, the destinations are approached in a sequential manner, without considering all possible orderings. We use a simple heuristic to order the destinations before planning. The destinations are ordered by the latest possible arrival time $t_d^{\mathsf{max}} - \delta_d$ so that the most urgent destination visits are performed first.

Another modification is the use of a reactive charging behavior. A typical user does not plan the charging until the battery has dropped below some threshold b_t which for

Schema: Worker				
#Dest.	Destination type	Time window	Dur.	$\|V_d\|$
1	Work	[8:00,18:00]	8h	1
2	Shopping	[7:00,21:00]	30min	1
3	Entertaining 1	[16:00,23:57]	1h	1
4	Entertaining 2	[16:00,23:58]	1h	1
5	Entertaining 3	[16:00,23:59]	1h	1

Schema: TSP				
#Dest.	Destination type	Time window	Dur.	$\|V_d\|$
1	Work 1	[8:00,18:00]	1h	1
2	Work 2	[9:00,19:00]	1h	1
3	Work 3	[10:00,20:00]	1h	1
4	Work 4	[11:00,21:00]	1h	1
5	Work 5	[12:00,22:00]	1h	1

Table 6.1: Temporal schemas for the generation of the problem instances.

the baseline algorithm is set to $b_t = 0.5 \cdot b_{\mathsf{max}}$. The charging is planned for each leg of the day plan separately.

6.3.2 Evaluation Problem Instances

As a testing area, we use a rectangular area of the real-world road network in Germany bounded by Munich, Regensburg and Passau with the transport network extracted from OSM[3] without local residential roads between cities leading to a graph with 75k nodes and 160k edges. We select 18 locations acting as possible POIs for the intermediate destinations and 8 of the 18 locations acting also as the charging stations. Each benchmark problem instance is generated based on one of the temporal schemas in Table 6.1 by randomly selecting a particular location for the intermediate destination. The most important aspect of each schema is the number of intermediate destinations, which range from 1 to 5. The variation between the number of intermediate destinations was achieved by taking only the first n destinations from the schema. For each schema and each number of intermediate destinations, we have generated 50 random instances (500 in total). All the planning requests start and end at the same location $v_{\mathsf{init}} = v_{\mathsf{goal}}$ with start time t_{init} set to 0 and with full battery $b_{\mathsf{init}} = b_{\mathsf{max}}$.

We used an EV model with 26kWh battery capacity and approx. 130km range with consumption modeled by function from Equation 3.2 with the following parameters: $\alpha^l = 0.2, \alpha^+ = 2, \alpha^- = 1.5$. Each charging station $q \in Q$ provides the same charging rate $P_q =$

[3]https://download.geofabrik.de/europe/germany/bayern.html

$50kW$. The *charging function* was simplified with linear approximation $\phi(b_{\mathsf{start}}, b_{\mathsf{end}}, P) = \frac{b_{\mathsf{end}} - b_{\mathsf{start}}}{P}$. The *charging waiting time* was set to zero $\tau_q(t, \delta^{\mathsf{ch}}) = 0$. The target charging SoC levels B were set to 20%, 40%, 60%, 80% and 100% of b_{max}.

6.3.3 HW and SW

We implemented the multi-destination EV travel planning algorithms in Java 8 and ran the experiments on a computer with Intel Xeon CPU E5-1650 v3 (3.5Ghz) and 64GB of RAM.

6.3.4 Whole Day vs. Single Trip Approach

In this experiment, we compare the baseline single-trip approach against our proposed holistic multi-destination approach based on a number of quality metrics. The first metric is the duration of the plan (i.e., makespan) including the travel times, times spent visiting destinations and time spent on charging if the charging is not performed in parallel while visiting destinations (in that case we take the maximum of the destination stop duration and of charging duration). The second metric is the consumption of electric energy (measured in kWh). The energy which was charged but not used for driving is not included. The last metric is the cost of the plan. We assume that the only cost comes from the charging and is proportional to the charging duration[4]. As in both our algorithms, the EV can be charged to only a set of predefined SoC levels, this may result in charging some energy that is not spent. This excess energy is also paid for and thus is included in the cost metric. Note that the algorithm proposed in Section 6.2 performs a single-objective optimization where the optimized metric is time only (that is, the duration of the plan).

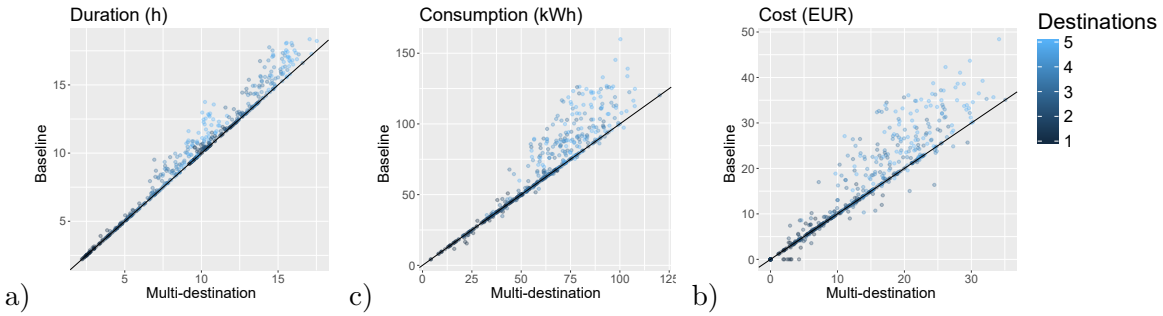
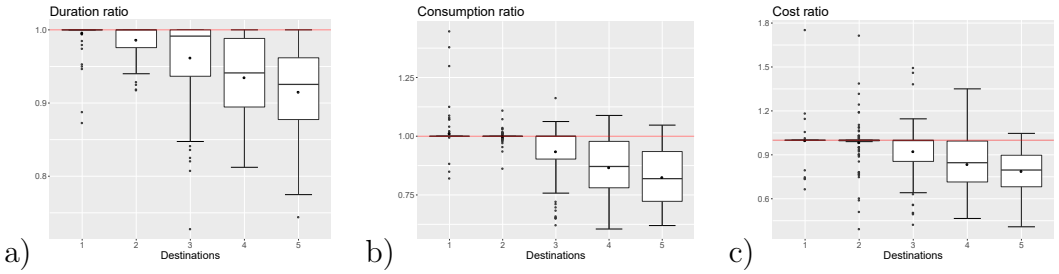


Figure 6.2: Ratios of the single-trip baseline and proposed global approach.

[4]At the time of experiments, most commercial charging stations used this type of pricing. Today, the situation is different and the cost is based on the amount of charged energy on 99% of charging stations in our most recent dataset (see Section 8.4.1).

Figure 6.2 shows the comparison for each of the considered metrics per problem for the proposed approaches (the x-axis) and the baseline approach as would be found by a human user (the y-axis). Let us first focus on the duration metric Figure 6.2(a) for which our proposed algorithm optimizes. Clearly, the optimized global approach is always better than the single trip baseline approach, sometimes with the difference in hours.

Somewhat unexpected are the results shown in Figure 6.2(b) and (c) which show that although the algorithm explicitly optimizes only for the time metric it outperforms the baseline solution in the two other metrics for most instances as well. This relates to the situation in the introductory example, where by optimizing the problem as a whole, future energy needs can be anticipated and detours necessary for charging can be eliminated.



a) b) c)

Figure 6.3: Ratios of the single-trip baseline and proposed global multi-destination approaches in dependence on the number of destinations in a problem.

In order to make a fair comparison, we evaluate the ratio of the proposed algorithm to the baseline approach for each metric. Figure 6.3 shows a boxplot[5] for each of the metrics. All three boxplots show that the more intermediate destinations are visited during the day, the bigger speed-up can be obtained from the multi-destination optimization. For five destinations, which is still a very reasonable number for an average user, the time spent on the plan may be more than 20% and on average nearly 10% shorter using multi-destination optimization. For an average 10h workday (including, e.g., shopping) this accounts for 2 and 1 hour respectively which is a very significant amount of time to be saved.

As already discussed, similar patterns can be observed for the metrics for which the algorithm does not explicitly optimize. Figure 6.3(b) shows that for five destinations a day, the proposed approach saves nearly 20% energy (and subsequently charging costs) on average.

[5]The boxplots show median (strong line), mean (black dot), the box showing Q1 (the 25th percentile) and Q3 (the 75th percentile) and the whiskers show the lowest and highest points within 1.5 IQR of the lower and higher quartile respectively. The outliers are shown as circles.

6.3.5 Heuristic and Pruning Evaluation

In this section, we evaluate the effect of the pruning speed-ups (Sections 6.2.1 and 6.2.2) and the heuristic (Section 6.2.3) on the performance of Algorithm 6.2 in terms of opened states which directly translates to the planning time. We decided to use the opened states as the main performance measure because it is independent of the experimentation environment and immune to measurement errors (no need to calculate the scenarios multiple times).

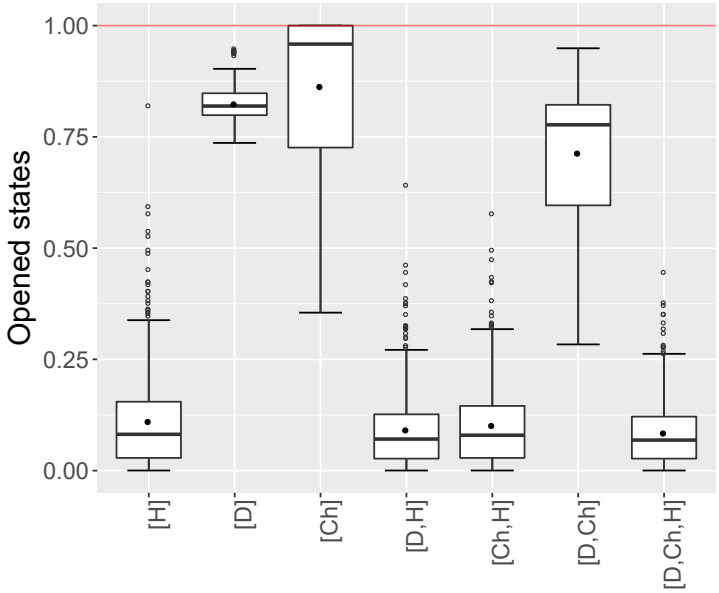


Figure 6.4: Opened states in dependence on used speed-ups relative to no speed-ups: H - heuristic (Section 6.2.3), D - Destination temporal consistency (Section 6.2.1), Ch - charging consistency (Section 6.2.2).

Figure 6.4 shows the comparison of ratios of the opened states of the proposed solution without any speed-ups and using the particular combination of speed-ups. The combinations of speed-ups excluding the heuristic perform significantly worse than the heuristic itself and any combination including it. The best result is obtained by combining all speed-ups which is not surprising. The combination of all reduces the solution time by 90% on average which shows that the proposed speed-ups are a significant improvement.

6.3.6 Effect of the Dominance Relaxation

This set of experiments evaluates the performance in terms of opened states and quality of the solution (the plan duration metric) when applying the dominance relaxation speed-up. In this experiment, all speed-ups from the previous section (pruners and heuristic) were used in combination with multiple settings of dominance relaxation described in Section 3.6.

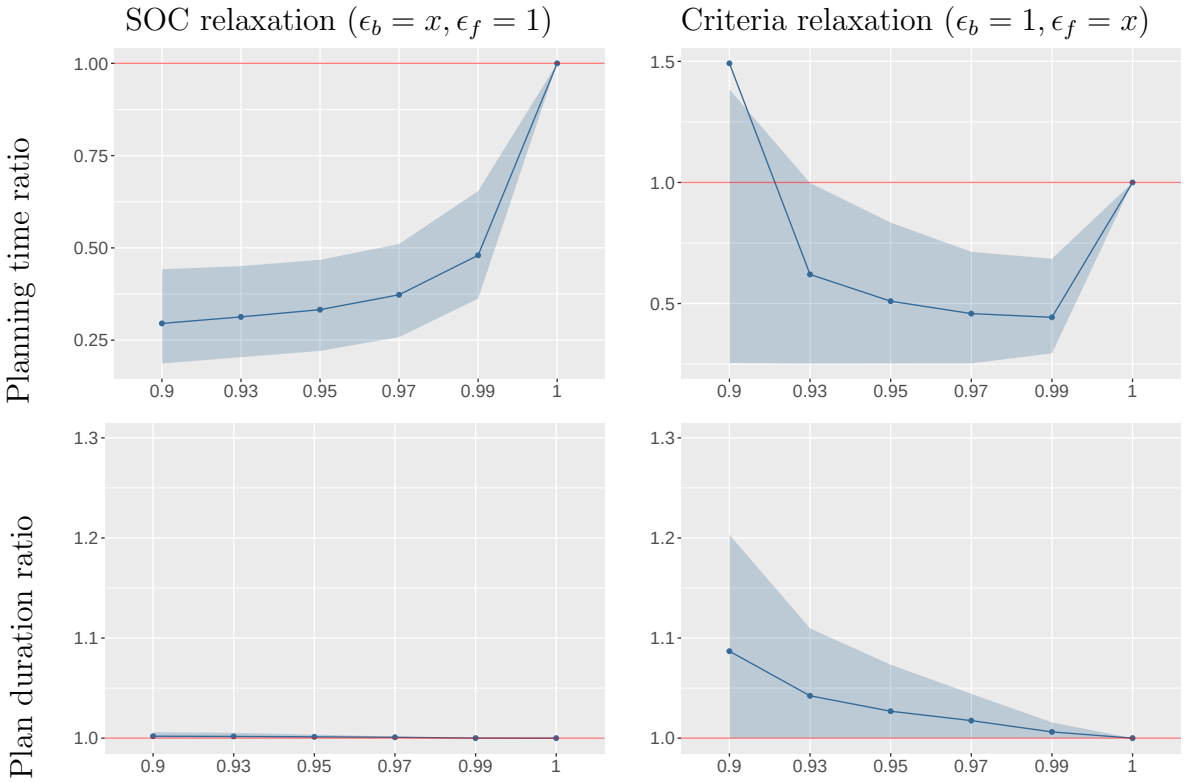


Figure 6.5: Effect of dominance relaxation on plan duration and algorithm planning time. The filled area represents data between the 1st and the 9th deciles.

Figure 6.5 shows the ratios of the algorithm using the given dominance relaxation value and using no dominance relaxation. The left column shows the SoC relaxation ϵ_b which relaxes the dominance only on the SoC whereas the right column shows the objective relaxation ϵ_t which relaxes the dominance on time attribute t. The results show that even though the number of opened states is reduced to 50% with SoC relaxation coefficient decreased from $\epsilon_b = 1$ to $\epsilon_b = 0.99$, the quality of the solution is practically intact.

As expected, there is a significant decrease in the number of states also with the time objective relaxation coefficient set to $\epsilon_t = 0.99$ but the impact on the plan duration is much greater than with SOC relaxation ϵ_b. The results suggest that the best trade-off between planning time and quality of the solution might be provided by the combination of $\epsilon_b = 0.99$ and $\epsilon_t = 0.99$. Indeed, such a combination reduces the planning time to 25% while not increasing the plan duration above 2.5% for 90% of the instances. An interesting result is that a lower time relaxation coefficient ϵ_t leads to a higher number of opened states. This is probably caused by pruning away too many labels resulting in the exploration of many detours that wouldn't be otherwise explored thanks to the heuristic.

6.4 Evaluation of Pre-Processing Speed-Up

The evaluation in the previous section was focused only on the multi-destination approach itself and the effect of the proposed speed-ups without pre-processing. In this section, we evaluate the memory and time requirements of the pre-processing speed-up (Section 6.2.4) and also the impact on query planning times. The pre-processing technique enables the algorithm to be used on larger problem instances. The problem instances used in the following evaluation are based on the ones described in Section 6.3.2 with several exceptions. We select 500-5000 random locations acting as possible POIs for the intermediate destinations and add 324 real-world charging station locations[6]. Each charging station is randomly assigned one of three charging rates (11kW, 30kW, and 50kW) and randomly parametrized waiting time which models morning and evening peak hours. Each benchmark problem is generated based on the temporal schema of 4-8 intermediate destinations by randomly selecting particular sets of locations for the destinations as shown in Table 6.2.

#Dest.	Destination type	Time window	Dur.	$\lvert V_d \rvert$
1	Work 1	[7:00,19:00]	4h	1
2	Work 2	[7:00,19:00]	4h	1
3	Shopping	[7:00,21:00]	0.5h	200
4	Other	[16:00,22:00]	1h	10
5 - 8	...	...	...	...

Table 6.2: Temporal schema used for the generation of problem instances for pre-processing evaluation.

At first, we evaluate the scalability of the pre-processing phase. Table 6.3 shows the relation of memory (in MB) and speed (in seconds) requirements with respect to $|V^{\mathsf{POI}}|$ for various pre-processing techniques where full is the full pre-processed set of all non-dominated routes, $\epsilon = 0.99$ uses the ϵ-pruning with given value of ϵ, and k uses the route reduction with given k. The results show, that the memory consumption grows approximately quadratically which corresponds with the expectations. The lowest memory footprint gives the route reduction with $k = 6$ (see Section 6.2.4.1). The ϵ-relaxation (Section 6.2.4.2) has a slightly larger memory footprint than $k = 6$, but smaller than $k = 10$ which suggests that the average number of u, v shortcuts $|E_{u,v}|$ for each $u, v \in V^{\mathsf{POI}}$ resulting from the ϵ-relaxation pruning technique lies between 6 and 10.

Regarding the pre-processing phase duration t_{pr}, we compare only the full set of routes and the ϵ-relaxation. For route reduction by clustering the pre-processing duration depends almost completely on the computation of full routes since the clustering duration

[6]The charging station locations are based on http://ev-charging.com

is negligible compared to the route computation. The results in Table 6.3 show that the pre-processing phase speed-up of ϵ-relaxation is more than 10× which is very significant.

$\lvert V^{\text{POI}}\rvert$	Memory [MB]					t_{pr} [min]	
	full	Route limit k			ϵ-rel.	full	ϵ-rel.
		20	10	6	0.99		0.99
824	620	255	149	95	107	70	5.3
1324	1640	666	388	247	282	119	8.6
1824	3190	1290	749	477	546	168	11.8
2324	5200	2098	1219	776	885	215	15.2
5324	-	-	-	-	4700	-	36.6

Table 6.3: Time and memory consumption of the pre-processing phase depending on $\lvert V^{\text{POI}}\rvert$ and on route reduction speed-up used.

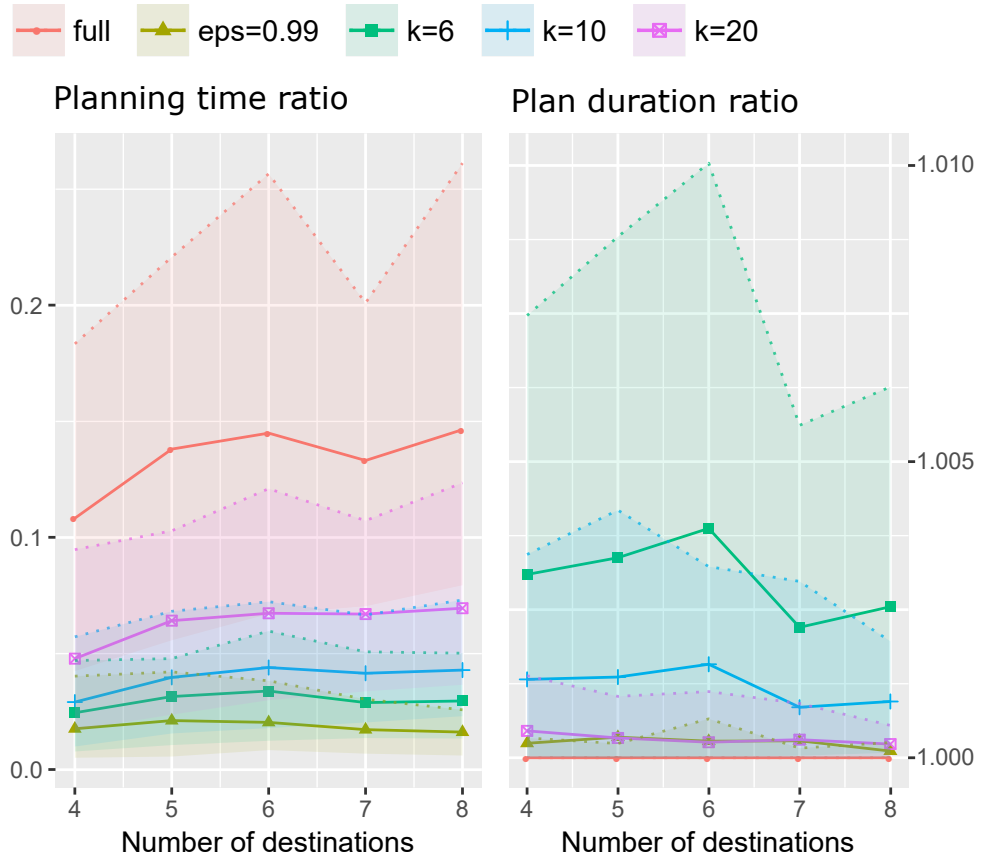


Figure 6.6: Comparison of the effect of pre-processing speed-ups on the planning time (left) and the quality (right). Values are average for 500 instances on 324 charging stations and 1000 POIs, the colored area corresponds to the region between the 1st and 9th deciles.

We also evaluate the performance of the whole algorithm and the quality of the solution depending on a number of factors. Let us first have a look at the effect of the pre-processing speed-ups. Figure 6.6 (left) shows the average ratio $t^{\text{Q}}/t^{\text{base}}$ of the time needed to find a solution where t^{Q} is the planning time of the pre-processing query phase (Section 6.2.4)

and t^{base} is the solution without pre-processing. The figure shows that the best solution time improvement comes from the ϵ-relaxation with less than 5% of solution time needed. Even the optimal variant with full set of routes provides more than 5× speed-up on average.

Figure 6.6 (right) shows the average ratio $t_\pi^{\mathsf{pre}}/t_\pi^{\mathsf{base}}$ of solution quality. Apart from the optimal full set of routes, the best quality solutions are provided by the ϵ-relaxation technique with solutions less than 0.1% worse than optimum which for a 12-hour plan corresponds to less than a minute.

We also experimented with limiting the number of charging stops between destinations (excluding charging at destinations). This is a reasonable constraint since most EV users would prefer a single charging stop instead of multiple ones even if it means a slightly longer plan. We measured the impact of the limit of 1 charging stop between each pair of destinations on the plan duration and the average delay is less than 5 seconds which is completely insignificant on a 12-hour plan.

However, the impact on the planning time is interesting. Figure 6.7 shows the query time improved approx. by 25% for both pre-processed variants. The figure shows the respective values only for the full set of routes (full) and the set of routes found using ϵ-relaxation with $\epsilon = 0.99$ (eps). The results clearly show that the query times of the ϵ-relaxation variant are below 5 seconds, which is a reasonable time. The full route set requires slightly longer running times but provides optimal solutions.

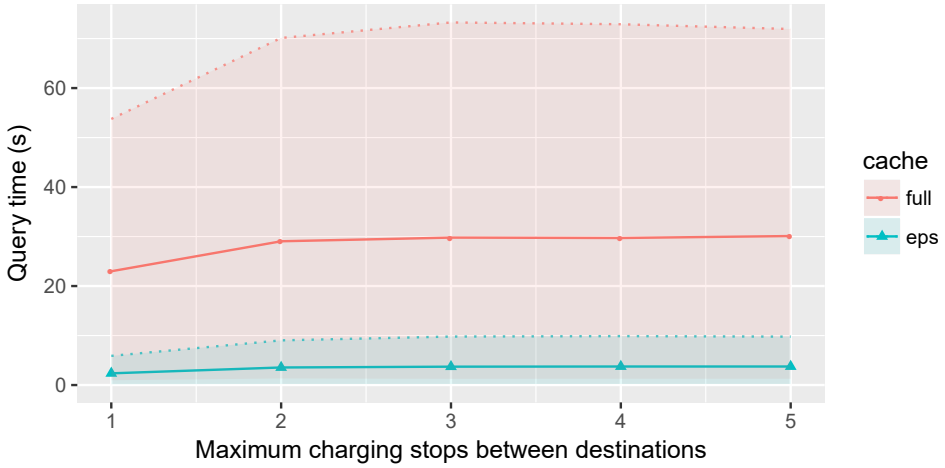


Figure 6.7: Average query times depending on the limit of charging stops between destinations. The colored area corresponds to the region between the 1st and 9th deciles.

6.5 Summary

In this chapter, we described the multi-destination variant of the EV travel planning problem and proposed an algorithm with several speed-up techniques to solve it.

The algorithm is built on the general multi-objective A* algorithm described in Chapter 5. We proposed a heuristic guiding the search and two pruning techniques to reduce the number of states to be explored. We also used the dominance relaxation technique that allows the algorithm to explore fewer states at the cost of slightly worse solution quality. We also proposed a pre-processing phase that pre-calculates the routes between all charging stations and points of interest.

Our evaluation proved that the holistic multi-destination approach significantly improves the results when compared to the baseline that just chains multiple single-destination plans. The travel plans are improved in terms of plan duration, energy efficiency, and also the overall cost even though the cost is not an optimization objective. We also evaluated the impact of the individual speed-up techniques and showed that they are crucial for the scalability of the algorithm.

While the pre-processing speed-up brings a significant improvement in the planning time, it also brings some restrictions. It may seem that the restriction that destinations must be from a pre-defined set of points of interest is crucial. Even though it would be most likely impossible to use it in large areas (because of memory requirements), it is useful for smaller areas such as the one we used for evaluation. Multi-destination planning is very useful even in these areas because most people regularly travel only within areas of such size. The experiments showed that the size of pre-processed data with 5 thousand POIs is approx. 5GB. This number of POIs should be sufficient for areas of the discussed size. For example, the area on which we evaluated the algorithm has approx. 15 000 km^2 and with 5 thousand POIs it has approx. 1 POI per 3 km^2. Which, if uniformly distributed, means that the maximum distance to the nearest POI from any point in the area is approx. 1km which is small enough to be handled by a post-processing. Moreover, we can decrease the distance by increasing the number of POIs which is still possible. 10 thousand POIs would require approx. 20GB of memory which is still feasible.

Chapter 7

Single-Objective Single-Destination EV Travel Planning with Incomplete Information

In this chapter, we focus on the uncertainty in the EV travel planning problem caused by incomplete information at the start of the search. The problem assumes that the information is not unknown entirely, but it is possible to gather it during the search. However, the receipt of the information is not instantaneous and takes some time. For example, the information can be gathered by a time-consuming computation or by a query from an external API.

More specifically, we focus on the single-objective single-destination EV travel planning problem where the waiting time τ_q on all charging stations $q \in Q$ has to be queried. Since the only objective is the total travel time we can dismiss all cost attributes of the problem. We can also dismiss the intermediate destinations.

This chapter is based on [Cuchý and Jakob, 2019].

7.1 Problem Definition

The single-objective single-destination EV travel planning problem with incomplete information is defined very similarly to the general problem definition $\mathcal{P} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R} \rangle$ with the following modifications:

- The solution to the problem is a single plan π optimized only for plan duration t_π.
- The *charging cost functions* γ_q and *cost per km of driving* ψ can be ignored.
- The states and all actions of EV travel plans can ignore the cost attribute.

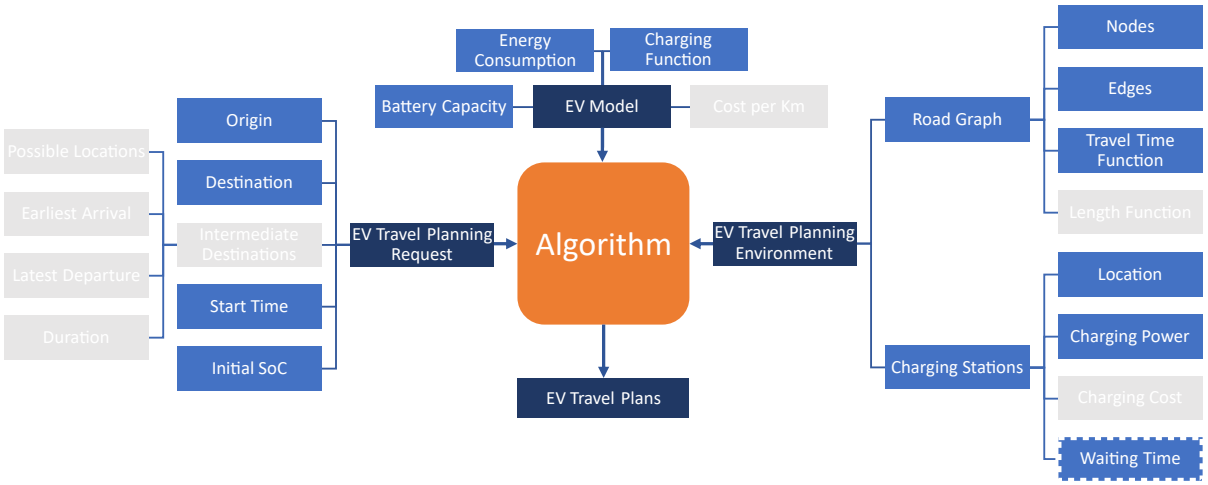


Figure 7.1: Schema of the single-objective single-destination EV travel planning problem with incomplete information. The greyed-out attributes are not used in this problem variant. The attribute with incomplete information has a dashed border.

- The states, all actions of EV travel plans and the planning request can ignore the intermediate destinations.
- The *waiting time* on charging stations τ_q is not known at the beginning of the search, but it is possible to query it during the search.

More specifically, the problem is defined by tuple $\mathcal{P}^{\mathsf{SOSD}} = \langle \mathcal{W}^{\mathsf{SO}}, \mathcal{M}^{\mathsf{SO}}, \mathcal{R}^{\mathsf{SD}} \rangle$ where $\mathcal{W}^{\mathsf{SO}} = \langle G, Q^{\mathsf{SO}} \rangle$ is a simplified (single-objective) environment, $\mathcal{M}^{\mathsf{SO}} = \langle b_{\mathsf{max}}, \beta, \phi \rangle$ is a simplified (single-objective) EV model, and $\mathcal{R}^{\mathsf{SD}} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ is a simplified (single-destination) request. Although the waiting time on charging stations τ_q is not known at the beginning of the search, the algorithm works with the estimate τ_q^* that is updated during the search.

The solution to the problem is a plan π optimized for plan duration t_π and for which all the waiting times are known accurately. It means that for each *charge(q,j)* action in the plan defined as:

$$\langle v_q, t, b \rangle \rightarrow \langle v_q, t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}, b + j \rangle$$

the waiting duration is known accurately $\delta^{\mathsf{w}} = \tau^*(t, \delta^{\mathsf{ch}}) = \tau(t, \delta^{\mathsf{ch}})$.

The schema of the problem with greyed-out attributes that are not used in this problem variant is shown in Figure 7.1.

7.2 Lazy Evaluation with Inference Algorithm

The lazy evaluation with inference algorithm for incomplete information EV routing problem works in two alternating phases or levels, similarly to the LazySP algorithm described by Dellin and Srinivasa [2016]. In the first phase, the lazy evaluation with inference algorithm searches for optimal plans according to the current state of information. In the second phase, the algorithm chooses which information to obtain and propagates (using inference) the retrieved information through the search space. The inference step which leverages prior domain knowledge (described below in detail) is the main difference from the LazySP algorithm and is expected to significantly reduce the amount of information required by the algorithm for the calculation of optimal plans.

In the lazy evaluation with inference algorithm a (sub)plan is represented as a tuple

$$s = \langle v, t, b, W \rangle$$

where $v \in V$ is the current node (location), $t \in \mathbb{R}_0^+$ is the time when the state is reached, $b \in [0, b_{\mathsf{max}}]$ is the current state of charge (higher value means more energy in the battery), and $W = (\langle q_0, t_0, \delta_0^{\mathsf{ch}} \rangle, \ldots, \langle q_k, t_k, \delta_k^{\mathsf{ch}} \rangle)$ is a sequence of triplets representing all charging actions that happened during the plan. Each triplet $\langle q, t, \delta^{\mathsf{ch}} \rangle$ consists of a charging station $q \in Q$, the earliest possible start time of the charging $t \in \mathbb{R}_0^+$ (arrival at the charging station), and the duration of the charging δ^{ch}.

7.2.1 First Phase: Route Planning

The lower level of the algorithm is a modification of the algorithm described in Section 5.3. It is a multi-objective algorithm that finds non-dominated plans that are optimal according to the minimal travel time and the maximum SoC at the destination. The algorithm works with the states defined in the previous section which have the following dominance relation:

Definition 13. Let $s = \langle v, t, b, W \rangle, s' = \langle v', t', b', W' \rangle$ be two states at the same node ($v = v'$). We say that s dominates s' (denoted as $s \preceq s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t &\leq t' \\ b &\geq b' \end{aligned} \tag{7.1}$$

For the algorithm to work, we need to define also the π-dominance which is almost the same except for the use of the heuristic function h_t. The used heuristic function $h_t(s) = \tau(v, v_{\mathsf{goal}})$ is the fastest travel time from the current node v to the destination

v_{goal}. It can be pre-calculated for all nodes by a backward Dijkstra's algorithm from the destination.

Definition 14. Let $s = \langle v, t, b, W\rangle, s' = \langle v', t', b', W'\rangle$ be two states. We say that s π-dominates s' (denoted as $s \preceq_\pi s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t &\leq t' + h_t(s') \\ b &\geq b' \end{aligned} \tag{7.2}$$

The states s in the opened set S^{op} are ordered first by time and a heuristic function $t + h_t(s)$ and then by SoC b. Let $s = \langle v, t, b, W\rangle$ be the minimal extracted state (Line 11 Algorithm 3). The `expand` step (Line 18 Algorithm 3) generates new states based on two actions:

(i) **move** For each outgoing edge $e = (v, u) \in E$, a new state

$$s = \langle u, t + \tau(e), \beta(e, b), W\rangle$$

is generated.

(ii) **charge** For each charging station $q = \langle v_q, P_q, \tau_q, \gamma_q\rangle$ such that $v_q = v$ and for each amount of energy $j \in J^B_{>b}$ a new state

$$s = \langle v_q, t + \delta^{\mathsf{ch}} + \tau^*_q(t, \delta^{\mathsf{ch}}), b + j, W \cup \langle q, t, \delta^{\mathsf{ch}}\rangle\rangle$$

is generated, where $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session, and $\tau^*_q(t, \delta^{\mathsf{ch}})$ is the estimate of waiting time before charging.

The result of this phase is an optimal plan according to the current state of knowledge represented by the waiting time estimate functions $\tau^*_q, \forall q \in Q$.

In general, the second phase of the algorithm requires the whole Pareto-set to be returned by the first phase to be able to choose the best query. However, some query selection strategies require only the fastest plan. In such a case, we can simplify the first phase algorithm to end immediately after finding the first plan.

7.2.2 Second Phase: Query Selection

In the main (second) phase, the algorithm is responsible for the selection of the information (waiting time) to be queried and the processing of the precise information gathered.

The pseudo-code of the second phase can be seen in Algorithm 4. The algorithm can be split into four repeated steps:

Algorithm 4: Pseudo-code of the main phase of the algorithm

```
1  Algorithm TwoPhasePlanner()
2      v_init: origin
3      v_goal: destination
4      t_init: start time
5      b_init: start energy
6      while true do
7          S ←routePlanning(⟨v_init, t_init, b_init, ∅⟩)
8          if s_best ∈ arg min_{s=⟨v,t,b,W⟩∈S} t is accurate then
9              return s_best
10         else
11             ⟨q, t, δ^ch⟩ ←select(S)
12             δ^w_acc ←query(q, t, δ^ch)
13             infer(q, t, δ^w_acc)
```

Route Planning At the beginning of the loop, the first phase of the algorithm finds non-dominated solutions S which are optimal according to the current knowledge of the waiting times time costs τ_q^*.

Selection In this step, the algorithm determines which charging action in found solutions $\langle q, t, \delta^{\mathsf{ch}} \rangle \in \bigcup_{\langle v,t,b,W \rangle \in S} W$, at what time t and with what duration δ^{ch} is worth a query for accurate information $\tau_q(t, \delta^{\mathsf{ch}})$. There are many possible strategies. However, in this thesis, we consider only the time-optimality-preserving strategy, which chooses the first unknown edge of the fastest plan s_{best} in S[1]. This strategy remains optimal if the estimate function τ_q^* is never greater for all charging stations q than the real cost τ_q – $\forall q \in Q; \forall t, \delta^{\mathsf{ch}} \in \mathbb{R}_0^+ : \tau_q(t, \delta^{\mathsf{ch}}) \geq \tau_q^*(t, \delta^{\mathsf{ch}})$ (similarly to optimality-preserving requirements on A* heuristic).

Query To get the accurate waiting time, a time-consuming query is made to the provider (for example, remote API).

Inference The waiting duration received $\delta^{\mathsf{w}}_{\mathsf{acc}} = \tau_q(t, \delta^{\mathsf{ch}})$ for charging station $q \in Q$, waiting start time t and charging duration δ^{ch} can be exploited to make the estimate function $\tau_q^*(t, \delta^{\mathsf{ch}})$ more precise for more input values than those used for the query.

Even if we arrive at the charging station after t, but before the end of the waiting $t_{\mathsf{end}} = t + \delta^{\mathsf{w}}_{\mathsf{acc}}$, the charging station will still be occupied. Similarly, it is impossible to charge at the charging station also for longer charging durations $\delta'^{\mathsf{ch}} \geq \delta^{\mathsf{ch}}$. If there is no free time slot for a short time interval, then there is also no time slots for a longer interval.

[1] Therefore, we can simplify the routing phase as mentioned above.

Therefore, we can update the estimate[2] $\forall t' \in [t - (\delta'^{\mathsf{ch}} - \delta^{\mathsf{ch}}), t_{\mathsf{end}}], \forall \delta'^{\mathsf{ch}} \in [\delta^{\mathsf{ch}}, t + \delta^{\mathsf{ch}}]$: $\tau_q^*(t', \delta'^{\mathsf{ch}}) = t_{\mathsf{end}} - t'$. These rules assume that the information provider always provides true information, e.g., if there is no booking at the given time, the provider will return the waiting time accordingly. It is possible that the provider will not allow, e.g., short charging sessions that could block longer and more profitable charging sessions. In that case, the inference has to be modified accordingly.

A solution is found if the time of the candidate plan with minimal time $\langle v_{min}, t_{min}, b_{min}, W_{min} \rangle \in \arg\min_{\langle v,t,b,W \rangle \in S} t$ returned from the route planning phase is accurate – $\forall \langle q, t, \delta^{\mathsf{ch}} \rangle \in W_{min} : \tau_q^*(t, \delta^{\mathsf{ch}}) = \tau_q(t, \delta^{\mathsf{ch}})$.

7.3 Evaluation

To evaluate the properties of the proposed algorithm and the properties of the incomplete information EV travel planing problem itself, we measure the influence of the parameters of the problem on the number of expensive queries required to find the optimal solution by the proposed algorithm. The basic parameter of the problem is the number of charging stations. Another important parameter is the density of the charging stations. It means, how many charging stations are relevant for specific routing requests. If there is the same number of charging stations in a larger area, the number of relevant charging stations decreases. The last parameter we focused on is the real waiting time function τ_q, more specifically the precision of the estimate τ_q^*.

We performed the evaluation on real-world based scenarios in the area of Germany and a smaller area of Bavaria in Germany. The road graph was extracted from OSM[3] and we randomly selected sets of 200, 500 and 1000 real-world charging stations for each area[4] and mapped them to the respective graphs. The road graphs were then pre-processed to contain only nodes with charging stations mapped to them and with edges representing the fastest routes between the charging stations. We used an EV model with 26kWh battery capacity and approx. 130km range with consumption modeled by function from Equation 3.2 with the following parameters: $\alpha^l = 0.2, \alpha^+ = 2, \alpha^- = 1.5$.

All charging stations are identical in terms of provided charging rates (50kW) and the the *charging function* was simplified with linear approximation $\phi(b_{\mathsf{start}}, b_{\mathsf{end}}, P) = \frac{b_{\mathsf{end}} - b_{\mathsf{start}}}{P}$. The target charging SoC levels B were set to 10%, 20%,..., 90% and 100% of b_{max}. The most important attribute of a charging station is its occupancy defining the waiting time τ_q. The process of how we generated the occupancy and thus the waiting time function

[2]The upper bound on δ'^{ch} is introduced only to prevent negative time t'

[3]https://download.geofabrik.de/europe/germany.html

[4]Charging stations locations downloaded from https://www.lemnet.org/

τ_q is described in detail in the next section.

7.3.1 Waiting Time Generation

The waiting time function τ_q is defined by the occupancy of the respective charging station. We define occupancy as a set of time intervals when the charging station is not available (essentially a reservation system). An illustration of how these time slots influence the waiting time function τ_q can be seen in Figure 7.2. We can see that the waiting time function is equal to zero unless there is a time interval during which the charging station is occupied.

The occupancy time intervals are generated individually for each charging station. In order to achieve the desired occupancy rate r (e.g. 20%), we need to generate a number of time slots. The number of slots n to be generated is sampled from a random normal distribution with $\mu_n = 24 \cdot r/\mu_\delta$ where μ_δ is the average duration of the slot. For each slot, we need to generate its midpoint and its duration. The duration δ is sampled from a normal distribution with the mean μ_δ at 30 minutes and the midpoint is sampled from a normal distribution with the mean at 11 a.m. truncated to interval $[\delta/2, 24 - \delta/2]$. The parameters of the distributions are based on real-world data[5] (the occupancy during a day can be seen in Figure 7.3). If the generated slot overlaps with one of the already generated slots, a newly generated slot is attempted for at most k times.

We generated the occupancy for ten occupancy rates (0%,10%,...,90%) and for each set of charging stations.

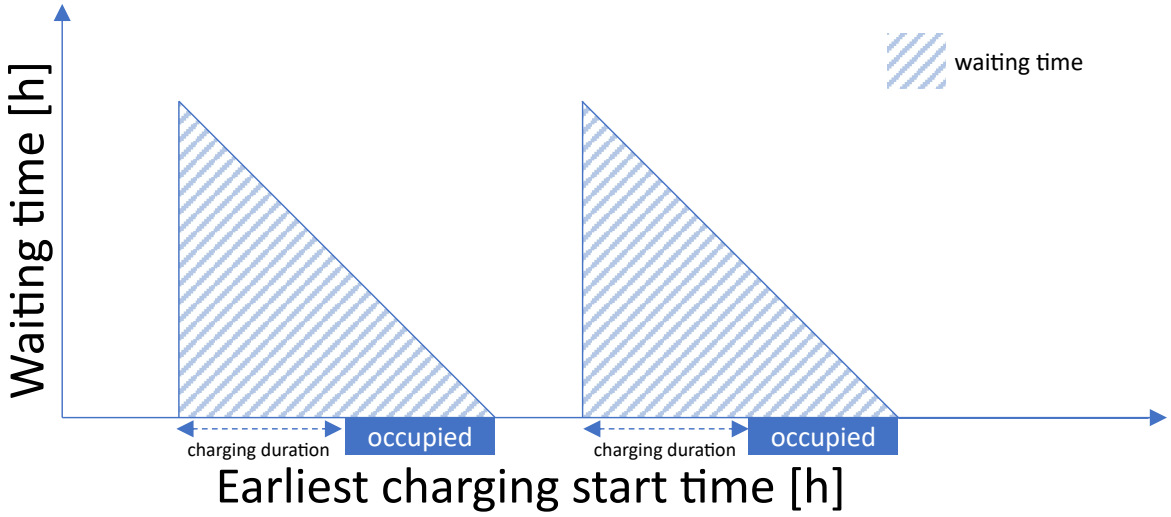


Figure 7.2: Illustration of waiting time function based on occupied time slots

[5]Data provided by E-WALD https://e-wald.eu/

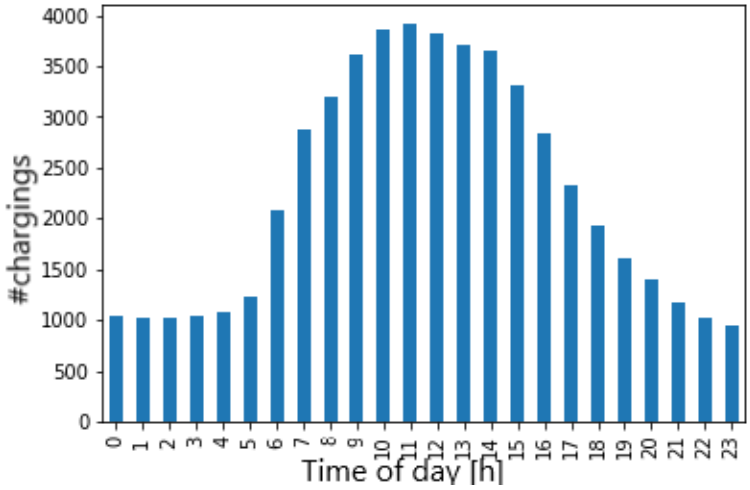


Figure 7.3: Charging occupancy during the day

7.3.2 Planning Request Generation

Each planning request $\mathcal{R}^{\mathsf{SD}} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ used in the evaluation is randomly generated. We randomly selected 1000 distinct origin-destination pairs of graph nodes for each of the six combinations of areas and numbers of CS (for each number combination there is a different graph because of the pre-processing mentioned above). For each pair, the start time (in hours) is sampled from a normal distribution with the mean at 8 a.m. The initial state of charge is always set to the maximum battery capacity.

7.3.3 Evaluation Results

We assume that the main influence on the problem complexity and number of expensive queries is the occupancy; therefore, we randomly generated the occupancy for 10 different occupancy rates r (0%, 10%,..., 90%) by the process described in Section 7.3.1 and calculated the plans for 1000 scenarios on two areas (Bavaria and Germany) and with 200, 500 and 1000 charging stations. We used the number of queries as the metric. The general overview of the results can be seen in Figure 7.4[6]. This figure confirms that the waiting time function, more precisely the occupancy, has a significant impact on the number of queries.

Judging from this figure, it may also seem as if the density of charging stations has only a small influence on the complexity ('Bayern' results need fewer queries). However, if we look at the comparison with the dependence on the length of the route in Figure 7.5 expressed as the number of charging sessions required to accomplish it, we can see that the density has a great impact too. The number of queries for larger areas most likely

[6]The glitch around 50% is probably caused by the implementation of the occupancy generation where it is switched from generation of occupied slots to generation of free slots

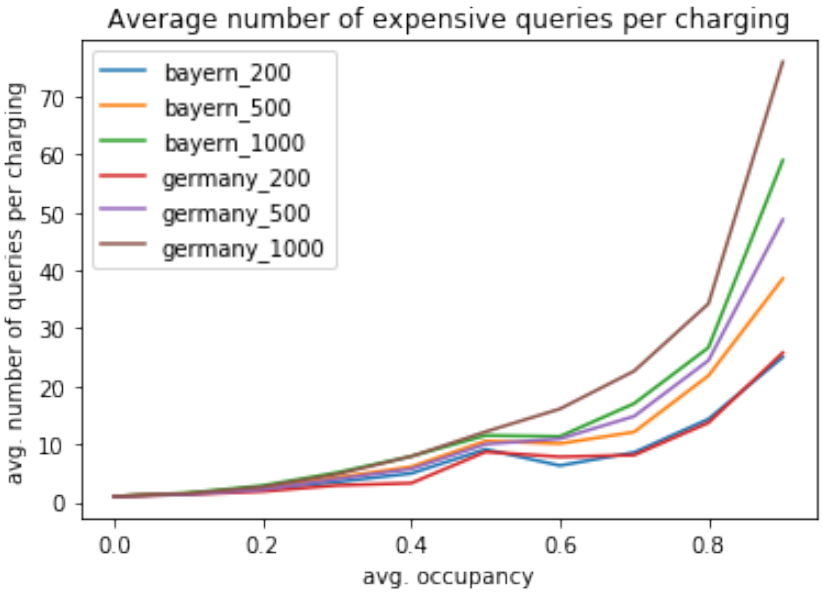


Figure 7.4: Average number of expensive queries per charging sessions considering average charging station occupancy.

increases because more trips require more charging sessions to reach the destination.

Moreover, in the context of EV travel planning, the impact of density is much more important due to the fact that routes with so many chargings during one trip will probably lead to the use of different means of transport. In addition, it could provide us space for decreasing the number of queries by selection of the charging stations in a more sophisticated manner considering the fact that there are probably more very similar alternatives and we could focus on identifying bottlenecks.

The results presented so far use the knowledge of the domain and leverage the inference of information from the queries. In Figure 7.6, we can see the benefits of this inference. With the increasing complexity of the problem (defined by avg. occupancy), the amount of queries saved by using the inference increases exponentially (as can be seen from the gap between the lines with and without inference - the gap is increasing even if the log scale is used).

7.4 Summary

In this chapter, we described the variant of the EV travel planning problem where the waiting time at charging stations is not known at the beginning of the search, but it is possible to gather it during the search.

We introduced an optimal algorithm with two inter-changing phases. The first phase, which is built on the general A* algorithm (Chapter 5), searches for the optimal travel plan based on the already available information. In the second phase, the algorithm

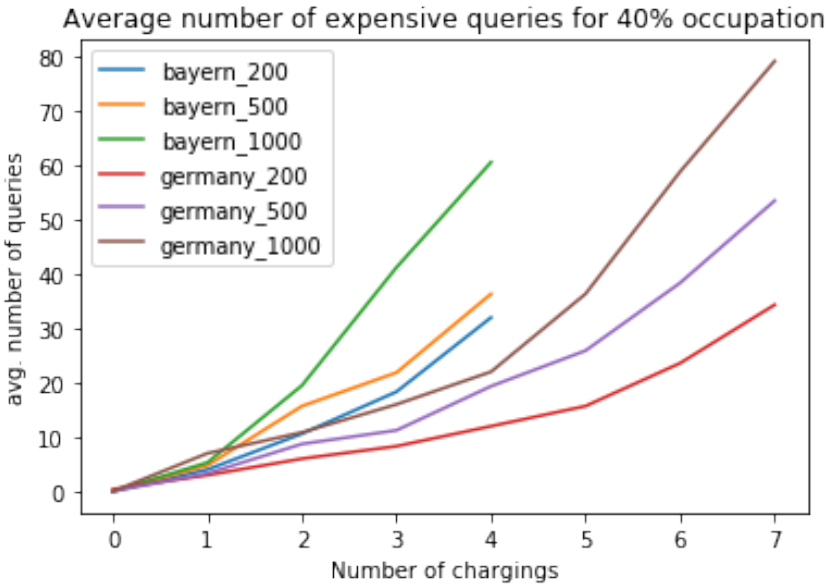


Figure 7.5: Average number of expensive queries with the dependence on the number of charging sessions. The small size of Bavaria leads to a smaller number of chargings required to get from one end to another; therefore, there are no plans with more than 4 charging sessions.

selects the information to be obtained and propagates it through the search space.

We have evaluated the properties of the problem on real-world scenarios in terms of their influence on the number of queries required during the search. We have shown that the complexity (measured as the number of queries) of the problem grows most likely exponentially with all considered parameters. In the context of EV and future work, the most influential parameters seem to be the density of charging stations and their occupancy. The length of the trip (number of chargings) is not so important given the fact that with the growing number of charging sessions, the usability of the EV itself rapidly decreases. We also showed that the information propagation via inference that was not used by previous approaches to similar problems can significantly reduce the number of queries.

Although the presented solution approach can be generalized and modified to other domains (e.g. planning with taxis or shared vehicles), the influence of the domain-specific inference from past queries is very significant.

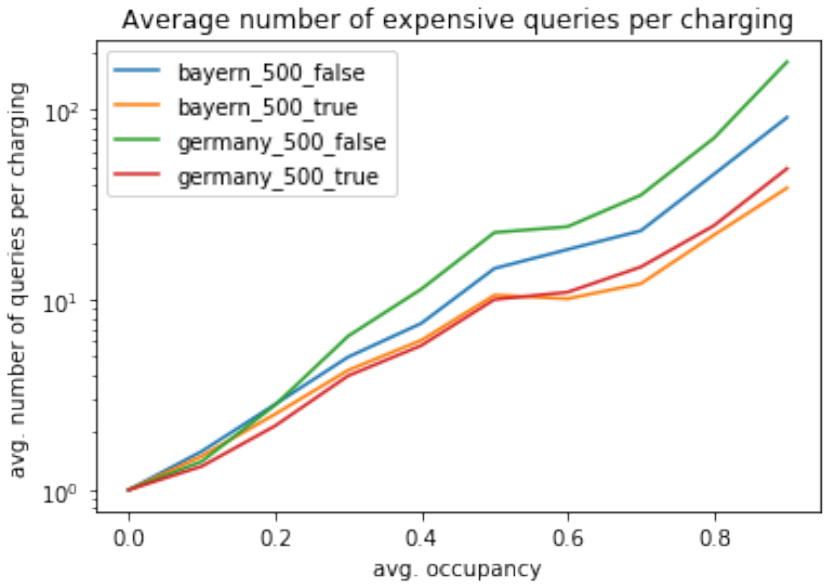


Figure 7.6: Influence of the inference on the number of queries. Notice the log scale. *true* - inference is used, *false* - planning without inference

Chapter 8

Multi-Objective Single-Destination EV Travel Planning

In this chapter, we describe the multi-objective single-destination sub-variant of the EV travel planning problem. This variant simplifies the problem by considering only the final destination and no intermediate destinations. Even without intermediate destinations, the problem is still challenging due to the multi-objective nature of the problem. It is NP-hard but not even in NP (details in Section 8.2).

We first studied the problem itself and its properties in [Cuchý et al., 2024a] and then enhanced the algorithm with contraction hierarchies pre-processing in [Cuchý et al., 2024b].

8.1 Problem Definition

The multi-objective single-destination EV travel planning problem is defined very similarly to the general problem definition $\mathcal{P} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R} \rangle$ with the following modification:

- The states, all actions of EV travel plans and the planning request can ignore the intermediate destinations.

More specifically, the problem is defined by tuple $\mathcal{P}^{\mathsf{MOSD}} = \langle \mathcal{W}, \mathcal{M}, \mathcal{R}^{\mathsf{SD}} \rangle$ where $\mathcal{W} = \langle G, Q \rangle$ is a planning environment, $\mathcal{M} = \langle b_{\mathsf{max}}, \beta, \phi, \psi, B \rangle$ is an EV model, and $\mathcal{R}^{\mathsf{SD}} = \langle v_{\mathsf{init}}, v_{\mathsf{goal}}, t_{\mathsf{init}}, b_{\mathsf{init}} \rangle$ is a simplified (single-destination) request. The schema of the problem with greyed-out attributes that are not used in this problem variant is shown in Figure 8.1.

8.2 Problem Complexity

In this section, we discuss the complexity of this problem variant. If we restrict the consumption to be always non-negative (i.e. we forbid recuperation), and remove all charging

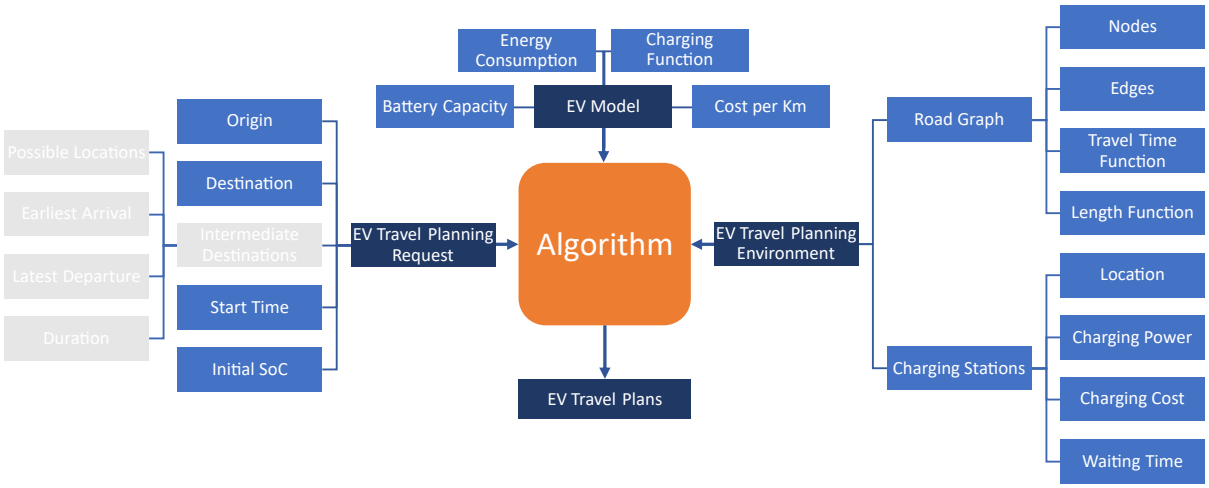


Figure 8.1: Schema of the multi-objective single-destination EV travel planning problem.

stations $Q = \emptyset$ and set *cost per km* to zero, then the multi-objective EV travel planning problem is equivalent to the NP-hard constraint shortest path problem [Garey and Johnson, 1979] and therefore the EV travel planning problem is also NP-hard. Moreover, since the bi-objective shortest path problem has in the worst case exponentially many plans in the solution Pareto-set [Müller-Hannemann and Weihe, 2006] given the length of the plan, the multi-objective EV travel planning problem, which is its extension, has also at least exponential size of the solution Pareto-set; this means the problem is at least in EXPSPACE.

8.3 Multi-Objective Single-Destination Algorithm

The algorithm is based on the multi-objective A* algorithm presented in Chapter 5. The algorithm uses simplified states and dominance relations, specific *state expansion* and two *heuristic* functions (Sections 8.3.1 and 8.3.2). We employed the dimensionality reduction technique (Section 8.3.3) proposed by Pulido et al. [2015] and dominance relaxation [Batista et al., 2011] described in Section 3.6. Furthermore, we also use the contraction hierarchies [Geisberger et al., 2012] pre-processing technique (Section 8.3.4) that leverages the hierarchical nature of road networks.

Since this problem does not consider intermediate destinations, the state can be simplified to $s = \langle v, t, c, b \rangle$. While π-dominance from Definition 8 remains almost unchanged[1] (see Definition 15), the more complex dominance from Definition 9 is modified according to the simplified states (see Definition 16).

[1] Only states are modified.

Definition 15. Let $s = \langle v, t, c, b\rangle, s' = \langle v', t', c', b'\rangle$ be two states. We say that s π-dominates s' (denoted as $s \preceq_\pi s'$) iff the following conditions are satisfied:

$$\begin{aligned} t &\leq t' + h_t(s') \\ c &\leq c' + h_c(s') \end{aligned} \tag{8.1}$$

Definition 16. Let $s = \langle v, t, c, b\rangle, s' = \langle v', t', c', b'\rangle$ be two states at the same node ($v = v'$). We say that s dominates s' (denoted as $s \preceq s'$) iff all the following conditions are satisfied:

$$\begin{aligned} t &\leq t' \\ c &\leq c' \\ b &\geq b' \end{aligned} \tag{8.2}$$

Let $s = \langle v, t, c, b\rangle$ be the minimal extracted state (Line 11 Algorithm 3) then the `expand` step is based on the two following actions:

(i) **move** For each outgoing edge $e = (v, u) \in E$, a new state

$$s = \langle u, t + \tau(e), c + \psi l(e), \beta(e, b)\rangle$$

is generated.

(ii) **charge** For each charging station $q = \langle v_q, P_q, \tau_q, \gamma_q\rangle$ such that $v_q = v$ and for each amount of energy $j \in J^B_{>b}$ a new state

$$s = \langle v_q, t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}, c + \gamma_q(t_q, j), b + j\rangle$$

is generated, where $\delta^{\mathsf{ch}} = \phi(b, b + j, P_q)$ is the duration of the charging session, and $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ is the waiting time before charging.

The `prune` function only checks if the battery constraints are satisfied. It would be also possible to use the pruning based on checking if a charging station is reachable before running out of energy as described in Section 6.2.2. However, the results discussed in Section 6.3.5 show that its impact on performance is very small even on small instances with only 8 charging stations. Its impact is even smaller with a realistic number of charging stations that dramatically decreases the distance to a nearest charging station and therefore decreases also the number of pruned states.

8.3.1 Remaining Travel Time Heuristic

This heuristic relaxes the battery constraints and estimates the minimum time needed to reach the destination regardless of the battery constraints. It calculates a lower bound on the travel time to the destination.

Let $s = \langle v, t, c, b \rangle$ be a state, then the heuristic can be expressed as

$$h_t(s) = t(v, v_{\mathsf{goal}})$$

where $t(v, v_{\mathsf{goal}})$ is the minimum travel time needed to drive from v to v_{goal}.

We pre-calculate the heuristic using a backward single-objective Dijkstra's algorithm.

Theorem 2. *The heuristic h_t is consistent.*

Proof. To prove the consistency of the heuristic, we need to prove the following condition is true for all consecutive states s, s':

$$t(v, v_{\mathsf{goal}}) \leq g(s, s') + t(v', v_{\mathsf{goal}}) \tag{8.3}$$

where g is the transition time cost between the two states. Note that the cost is non-negative.

In the case of the *move* action, the condition is always true since it is essentially the triangle inequality, which the shortest path satisfies.

Now we prove the consistency condition also holds for *charge* action. We prove this by contradiction. If any charge action violated the condition, there would have to exist a charge action (defined by the two states s, s') such that: $t(v, v_{\mathsf{goal}}) > g(s, s') + t(v', v_{\mathsf{goal}})$. During a charge action the location does not change ($v = v'$), therefore the heuristic also does not change: $t(v, v_{\mathsf{goal}}) > g(s, s') + t(v, v_{\mathsf{goal}}) \iff 0 > g(s, s')$. Since the transition cost cannot be negative, we have a contradiction. Therefore, all *charge* actions satisfy the consistency condition.

There are no transitions other than *move* and *charge* actions. Therefore, all transitions satisfy the consistency condition.

□

8.3.2 Minimum Remaining Charging and Driving Cost Heuristic

Since the cost objective comprises two components - the charging cost and the driving cost - the heuristic is based on the combination of the lower bounds of both individual components. The calculation of the minimum cost spent on charging is based on the most energy-efficient route to the destination, while the minimum driving cost is based on the length of the shortest route.

Let $s = \langle v, t, c, b \rangle$ be a state, then the heuristic can be expressed as

$$h_c(s) = b_{\mathsf{min}} c_{\mathsf{min}} + \psi l(v, v_{\mathsf{goal}})$$

where b_{min} is the minimum amount of energy that has to be charged to reach the destination (details below), c_{min} is the minimum possible price per amount of energy achievable with regards to the cost functions of all charging stations and the charging function of the EV, and where $l(v, v_{\mathsf{goal}})$ is the length of the shortest path from v to v_{goal}. The minimum amount of energy that has to be charged to reach the destination $b_{\mathsf{min}} = \beta(v, v_{\mathsf{goal}}) - b$ is the amount of energy required by the most energy efficient route from v to v_{goal} deducted by the current SoC b.

The minimum price per amount of energy c_{min} calculation cannot be easily described for the general case since both the charging function and the cost function consider too many parameters. However, it is possible to do so for a specific case. For example, suppose the cost function is duration-based (for instance, per minute of charging), and the charging function is piecewise linear. In that case, we can find the segment of the charging function where the charging function is the most efficient and compute the minimum price per unit of energy based solely on the most efficient segment of the charging function. Calculation of the minimum price is also simple for the fixed price per charging session, i.e., the same cost no matter how much energy is charged or how long the charging takes. In such a case, the minimum price per amount of energy is achieved while charging the battery from complete depletion to the maximum capacity. Therefore, we divide the fixed price by the battery capacity.

We pre-calculate the heuristic by a backward label-correcting version (due to the negative consumption) of single-objective Dijkstra's algorithm.

Theorem 3. *The heuristic h_c is consistent.*

Proof. To prove the consistency of the heuristic, we need to prove the following condition is true for all consecutive states s, s':

$$(\beta(v, v_{\mathsf{goal}}) - b)c_{\mathsf{min}} + \psi l(v, v_{\mathsf{goal}}) \leq g(s, s') + (\beta(v', v_{\mathsf{goal}}) - b')c_{\mathsf{min}} + \psi l(v', v_{\mathsf{goal}}) \quad (8.4)$$

where g is the transition money cost between the two states. Note that the cost g and the minimum price c_{min} are non-negative.

We again split the proof based on the two possible actions: *move* and *charge*. To prove the consistency for all *move* actions, we split the original inequality condition 8.4 into two separate inequalities (one for the charging cost part and one for the driving cost

part) that if both are satisfied, also the original inequality is satisfied:

$$\psi l(v, v_{\mathsf{goal}}) \leq g(s, s') + \psi l(v', v_{\mathsf{goal}}) \tag{8.5}$$

$$\wedge$$

$$(\beta(v, v_{\mathsf{goal}}) - b)c_{\mathsf{min}} \leq (\beta(v', v_{\mathsf{goal}}) - b')c_{\mathsf{min}} \tag{8.6}$$

$$\Longrightarrow$$

$$(\beta(v, v_{\mathsf{goal}}) - b)c_{\mathsf{min}} + \psi l(v, v_{\mathsf{goal}}) \leq g(s, s') + (\beta(v', v_{\mathsf{goal}}) - b')c_{\mathsf{min}} + \psi l(v', v_{\mathsf{goal}})$$

Since all *move* actions cost $g(s, s') = \psi l(v, v')$, the inequality 8.5 is equivalent to: $\psi l(v, v_{\mathsf{goal}}) \leq \psi l(v, v') + \psi l(v', v_{\mathsf{goal}})$. If the cost per km $\psi = 0$, the inequality is trivially true. If $\psi > 0$, the inequality simplifies to $l(v, v_{\mathsf{goal}}) \leq l(v, v') + l(v', v_{\mathsf{goal}})$, which is the triangle inequality for shortest path and therefore satisfied. In the case the minimum price c_{min} in inequality 8.6 is 0, the condition is trivially true. In the case the minimum price is positive, the inequality simplifies to: $\beta(v, v_{\mathsf{goal}}) - b \leq \beta(v', v_{\mathsf{goal}}) - b' \iff \beta(v, v_{\mathsf{goal}}) \leq \beta(v', v_{\mathsf{goal}}) + b - b'$. Since the SoC difference $b - b'$ can be seen as the energy transition cost between the two states (and one edge in the graph), the last inequality is essentially the triangle inequality with the most energy-efficient paths. Even though the energy cost of an edge can be negative, there are no negative cycles; therefore, the inequality is still satisfied, and therefore, the inequality 8.6 is true for all *move* actions. Since both inequalities (8.5 and 8.6) are true for all *move* actions, also the original inequality condition 8.4 is true for all *move* actions.

During a *charge* action, the location does not change ($v = v'$), therefore: $(\beta(v, v_{\mathsf{goal}}) - b)c_{\mathsf{min}} + \psi l(v, v_{\mathsf{goal}}) \leq g(s, s') + (\beta(v, v_{\mathsf{goal}}) - b')c_{\mathsf{min}} + \psi l(v, v_{\mathsf{goal}}) \iff \beta(v, v_{\mathsf{goal}})c_{\mathsf{min}} - bc_{\mathsf{min}} \leq g(s, s') + \beta(v, v_{\mathsf{goal}})c_{\mathsf{min}} - b'c_{\mathsf{min}} \iff (b' - b)c_{\mathsf{min}} \leq g(s, s')$. Since c_{min} is the minimum achievable price per energy among all the charging stations and $b' - b$ is the amount of charged energy during the charge action, the cost of the charge action $g(s, s')$ cannot be smaller, and the inequality holds.

□

8.3.3 Dimensionality Reduction

The greatest bottleneck of our proposed algorithm is the computational complexity of dominance checks that is directly dependent on the size of the Pareto-sets managed by the algorithm ($S_v^{\mathsf{op}}, S_v^{\mathsf{cl}}$, and Π). The size of the Pareto-sets can grow exponentially with the size of the problem (in particular, with the size of the road graph and the number of charging stations) and the number of components on which the dominance is based, making the dominance checks very expensive.

Fortunately, we can leverage a technique proposed by Pulido et al. [2015] that reduces the dimension of some of the Pareto-sets without loss of optimality. If we use the lexicographical ordering for the minimal label $s_{\min}$ extraction (line 11 in Algorithm 3) and if the heuristic estimates h_t and h_c are consistent, we can remove the first attribute (in our case the time) from the dominance checks against the solution set Π (lines 12 and 21) and against the closed set S_v^{cl} (line 21). Unfortunately, it does not apply to the opened set S_v^{op}.

The core idea is that if the algorithm extracts the best states at first by time (lexicographical ordering) and if the heuristics are consistent, we know that the closed and solution sets cannot contain states that are worse in the time attribute than the states extracted later. Pulido et al. [2015] prove the optimality of the technique if the above-mentioned requirements are satisfied.

8.3.4 Contraction Hierarchies

Contraction hierarchies (CH) [Geisberger et al., 2008] are a well-known pre-processing technique that reduces the complexity of the route planning part of the problem. Baum et al. [2019a] use them similarly to us.

Contraction hierarchies (or more general highway hierarchies) leverage the hierarchical nature of road networks. A vehicle trip commonly starts at a road of local importance, for example, a tertiary road, and then goes up to a secondary road, then to a primary road, and eventually ends on a highway.

Contraction hierarchies work in two phases. The pre-processing phase assigns a level $lvl(v)$ to each node $v \in V$ and calculates shortcuts E^{CH} that speed up the query phase. The query phase then performs a search on the graph enhanced with the shortcuts $G^{\mathsf{CH}} = \langle V, E \cup E^{\mathsf{CH}} \rangle$ limited only to *up-down* paths. An up-down path is a path where the level of the nodes is non-decreasing at the first part of the path and decreasing at the rest of the path.

For the contraction hierarchies to work in our case, they must be extended to the multi-objective setting. More specifically, we need the shortcuts to be Pareto-optimal with regard to the time, distance and SoC profile defining energy consumption. The distance is required because the cost objective comprises wear-and-tear costs that are directly dependent on the distance. The SoC profile is required because the SoC at the start of the shortcut is not known at the time of the pre-processing.

8.3.4.1 Pre-Processing Phase

In the pre-processing phase, the nodes of graph G are contracted one by one. When a node $v \in V$ is contracted, for each pair of incoming edge $(u, v) \in E$ and outgoing edge $(v, w) \in E$, a shortcut $e' = (u, w)$ is calculated by their concatenation. The contracted node and its adjacent edges are then removed from the graph. Afterwards, for each shortcut, a *witness search* is started. A *witness search* determines if there exists a *witness* path that dominates the shortcut. If a *witness* path exists, the shortcut is not needed and, therefore, discarded since there exists a better/dominating path. To improve the performance, we calculate the witness search at once for all shortcuts starting at the same node u by a version of multi-objective Dijkstra's algorithm very similar to the algorithm described in Algorithm 2. We also use *hop limit* that bounds the search only to the vicinity of the origin (in our case, to paths consisting of 20 edges at maximum). Although it leads to the addition of unnecessary edges, it does not violate optimality.

The next vertex to contract is determined based on a priority composed of three node metrics proposed by Geisberger et al. [2012] - Edge Difference (ED), Cost of Queries (CQ) and Deleted Neighbors (DN). The resulting priority is $64\mathsf{ED} + \mathsf{CQ} + \mathsf{DN}$ as used by Baum et al. [2019a]. The priority is calculated once for all nodes at the beginning of the pre-processing and stored in a priority queue. Furthermore, we implemented a lazy update of the priority. When a node with minimal priority is polled from the queue, its priority is recalculated, and if the priority is higher than the second smallest priority, the node is reinserted into the queue. The process is repeated until the priority of a node remains the smallest after its update. Additionally, we update the priority of all neighbors of a contracted node. This can easily be done in parallel since they do not change anything until the queue is updated, which can be done in a serial manner after all priorities are calculated. The resulting *contraction order* defines the level of the contracted nodes.

It is not required for all nodes to be contracted. It is commonly used in more complex scenarios [e.g., Baum et al., 2019a] to, for example, lower the number of created shortcuts If there are too many of them, it could negatively impact the query performance. In our case, we also need it to allow traveling between charging stations required by the need for charging. The set of uncontracted nodes $V^\circ \subset V$ is called the *core* and contains at least all nodes with charging stations $V_Q \subseteq V^\circ$. All nodes in the core have assigned equal level $\forall v \in V^\circ : lvl(v) = |V| - |V^\circ| + 1$.

An edge (u, v) is an *upward* edge iff $lvl(u) \leq lvl(v)$ and *downward* edge iff $lvl(u) > lvl(v)$. An *upward* graph $G^\uparrow = \langle V, E^\uparrow \rangle$ is a graph where all edges $E^\uparrow \subset E \cup E^{\mathsf{CH}}$ are upward while *downward* graph $G^\downarrow$ contains only downward edges $E^\downarrow$.

The edges $e \in E$ in the original graph G of the problem $\mathcal{P}$ have defined three properties - traversal duration $\tau(e)$, distance $l(e)$, and energy consumption function $\beta(e, b)$ which is

part of EV model $\mathcal{M}$ and represents SoC profile described in Section 3.7. The duration and distance of a shortcut created by a concatenation of two edges are trivial. Since the consumption functions are SoC profiles, their concatenation follows the rules described in Section 3.7. It has also defined dominance relation and therefore allows to easily check the dominance of shortcuts and found paths by, e.g., witness search.

8.3.4.2 Query Phase

CH queries are commonly solved by bidirectional search algorithms. However, our problem is too complex for easy adoption of bidirectional search, mostly because of the time-dependent nature of charging (dependence on starting SoC) that makes backward search that includes charging very complicated. Therefore, we split the query phase into two sub-phases similarly to [Baum et al., 2019a].

First, we run a backward search starting at the destination v_{goal} on the downward contracted graph $G^{\downarrow}$ that calculates temporary shortcuts E^{dest} from the uncontracted *core* (that contains all charging stations) to the destination. This search is based on multi-objective Dijkstra's algorithm very similar to the algorithm used by the witness search in the pre-processing phase. Since the SoC is unknown at the time of the calculation, the algorithm calculates the *SoC profile* instead of just the consumption values.

The second sub-phase runs the multi-objective A*-based Algorithm 3 described in Chapter 5 and specialized in Section 8.3 on the upward graph $G^{\uparrow}$ with added temporary shortcuts E^{dest} from the first sub-phase instead of the original road graph G: $\overline{G} = \langle V, G^{\uparrow} \cup E^{\mathsf{CH}} \rangle$. The rest of the input remains the same.

The query phase as described above maintains optimality of the underlying A*-based algorithm. The query phase limits the search only to *up-down* paths[2] and CH guarantee that if a Pareto-optimal path exists in the original graph G, then an *up-down* path with the exact same costs also exists in the contracted graph G^{CH}. The claim is a generalization (from shortest path to Pareto-optimal path) of a claim by Geisberger et al. [2012] which can be proved by a simple and straightforward generalization of the proof of the original claim.

8.3.5 Algorithm Properties

In this section, we examine and prove the properties of the above-proposed algorithm. First, we focus on the so-called *uninformed algorithm*, which is a simplified version of the algorithm described above without any speed-ups and heuristics. The uninformed algorithm is basically a multi-objective extension of Dijkstra's algorithm. After that,

[2] *Up path* - connects origin to the CH core, *equal path* - moving accross the core (between charging stations), *down path* - the end segment that connects the core with the destination.

we show that the heuristics, dimensionality reduction and contraction hierarchies do not affect the optimality of the algorithm.

Uninformed Algorithm Properties

At first, we prove that all the plans/states it finds are non-dominated by each other. Then, we prove that it finds all plans that are Pareto-optimal with regard to the problem definition and the restriction posed on the charging actions by the predefined target charging levels. Finally, we prove that the algorithm terminates in a finite number of steps.

Theorem 4. *The uninformed algorithm has the label/state setting property.*

The label/state setting property means that when a state is extracted (settled) from the priority queue S^{op}, it will not be dominated by any other state at the same node that is extracted from the queue later. That means that the state is Pareto-optimal with regard to all other states at the same node.

Proof. Assume that the extracted state $s = \langle v, t, c, b \rangle$ will be dominated. It means that a state $s' = \langle v', t', c', b' \rangle$ at the same node ($v' = v$) with all the following properties has to appear in the queue: lower or equal time ($t' \leq t$), lower or equal cost ($c' \leq c$), and higher or equal SoC ($b' \geq b$). State at the same node with all attributes equal cannot appear in the queue since it would be dominated by s in the closed set S_v^{cl} and pruned (line 21 in Algorithm 3). Therefore, state s' has to be strictly better at at least one of the attributes. Since state s is lexicographically smallest in the queue, state s' cannot be in the queue and is yet to be generated. It means that state s' has to be generated from a state $s'' = \langle v'', t'', c'', b'' \rangle$ that is already in the queue, or from the extracted state s itself (creating a cycle), or from any of their successors[3]. We will prove that any state generated from s'', s, or any of their successors cannot dominate s. State s'' is lexicographically greater or equal to the lexicographically smallest label s. Therefore, it has:

1. higher time ($t'' > t$), or
2. equal time ($t'' = t$) and higher cost ($c'' > c$), or
3. equal time ($t'' = t$), cost ($c'' = c$), and smaller or equal SoC ($b'' \leq b$).

The first two cases cannot generate a state that dominates state s, since both time and cost objectives are non-decreasing. In the third case, it is possible to improve the state of charge (by recuperation or charging), but it is impossible without increasing the

[3]We call a newly generated state a *successor* of the original state. It has the transitive property - meaning that a successor of a successor is also a successor of the first state.

time objective since it is impossible to move or charge in no time. The third case also applies to the expansion of the extracted state itself (everything is equal), therefore, also in this particular case a dominating state cannot be generated. Following all of the above, the successors of s'', or s cannot be lexicographically smaller than s. Therefore, also the successors cannot generate a state dominating s.

Therefore, it is impossible to add a state dominating s to the queue. □

Since any state extracted from the priority queue is Pareto-optimal with regards to the states at the same node, the extracted states at the destination node are also Pareto-optimal.

Lemma 1. Let a sub-plan π_i^* of a Pareto-optimal plan $\pi^* = (s_0, a_0, s_1, a_1, \ldots, a_{k-1}, s_k)$ be its sub-sequence of states and actions $(s_0, a_0, s_1, a_1, \ldots, a_{i-1}, s_i)$, where $i < k$. Each sub-plan π_i^* of a Pareto-optimal plan π^* is also Pareto-optimal with regards to all (sub-)plans ending at node v_i.

Proof. We will prove the Pareto-optimality of sub-plans by proving that it is impossible to generate a Pareto-optimal (sub-)plan from a dominated (sub-optimal) sub-plan, therefore, it is also impossible to generate a Pareto-optimal solution plan π^* from a dominated sub-plan.

Assume that $s = \langle v, t, c, b \rangle$ is the end state representing the dominated sub-plan, which means that there exists a state (representing another sub-plan) $s' = \langle v', t', c', b' \rangle$ such that $s' \preceq s \wedge v = v'$. We claim that there does not exist an action a applied to s that would generate a state $s_a = \langle v_a, t_a, c_a, b_a \rangle$ that would not be dominated by a state $s'_a = \langle v'_a, t'_a, c'_a, b'_a \rangle$ generated by the same action applied to s' or already by the state s' itself. We prove it for *move* actions and separately for *charge* actions.

The above holds for all *move* actions a because they satisfy FIFO[4] property since they are defined solely by additions and subtraction of constant values defined by the road graph edges[5]. Therefore, it is impossible to achieve non-dominated state attributes with dominated initial state by a move action ($s' \preceq s \implies s'_a \preceq s_a$).

The *charging* actions are generated for the predefined set of target charging levels B. We can divide the charging actions in two cases by the considered target charging level $b^{\mathsf{end}} \in B$:

Case 1: $b < b^{\mathsf{end}} \leq b'$ Since the charging function ϕ is positive and $t \geq t'$ (from the definition of dominance), then $t_a > t'$. Since all charging cost functions γ_q are non-negative and $c \geq c'$, then $c_a \geq c'_a$. And since $b_a = b^{\mathsf{end}} \leq b'$, then $s' \preceq s_a$.

[4]First-in first-out: a worse input value cannot generate a better result. For example, a later departure cannot lead to earlier arrival.

[5]The battery capacity maximum in SoC calculation does not invalidate the FIFO property

Case 2: $b^{\mathsf{end}} > b'$ We define the amount of charged energy for the dominated state s as $j = b^{\mathsf{end}} - b$ and for the dominating state s' as $j' = b^{\mathsf{end}} - b'$ while $b^{\mathsf{end}} = b_a = b'_a$. We also denote the charging durations as $\delta^{\mathsf{ch}} = \phi(b, b^{\mathsf{end}}, P)$ and $\delta'^{\mathsf{ch}} = \phi(b', b^{\mathsf{end}}, P)$, and the charging waiting durations at charging station $q \in Q$ as $\delta^{\mathsf{w}} = \tau_q(t, \delta^{\mathsf{ch}})$ and $\delta'^{\mathsf{w}} = \tau_q(t', \delta'^{\mathsf{ch}})$.

Since $b \leq b'$, then $j \geq j'$. When a target charging level (and charging power P) is equal, the charging function ϕ satisfies the FIFO property[6]. Therefore, $\delta^{\mathsf{ch}} \geq \delta'^{\mathsf{ch}}$. Charging waiting time τ also cannot return a shorter waiting duration if the charging start time is not smaller and charging duration is also not smaller: $t \geq t' \wedge \delta^{\mathsf{ch}} \geq \delta'^{\mathsf{ch}} \implies \delta^{\mathsf{w}} \geq \delta'^{\mathsf{w}}$. And since, $t \geq t'$ and $t_a = t + \delta^{\mathsf{ch}} + \delta^{\mathsf{w}}$ (accordingly for t'_a) , then $t_a \geq t'_a$.

Since both charged energy and charging duration are greater for the dominated state s, none of the charging cost functions γ_q can yield a cheaper charging ($\forall q \in Q : \gamma_q(\delta^{\mathsf{ch}}, j) \geq \gamma_q(\delta'^{\mathsf{ch}}, j')$; and since $c \geq c'$, then $c_a \geq c'_a$.

We proved that $t_a \geq t'_a$ and $c_a \geq c'_a$ and because $b_a = b'_a = b^{\mathsf{end}}$, then $s'_a \preceq s_a$. □

Definition 17. Let Π^* be the non-dominated set of all plans solving the given problem instance: there does not exist a plan $\pi \notin \Pi^*$ ending in the destination such that $\nexists \pi^* \in \Pi^* : \pi^* \preceq \pi$.

Theorem 5. *The uninformed algorithm finds* Π^*.

Proof. Since all sub-plans of a Pareto-optimal solution plan are Pareto-optimal (Lemma 1), they cannot be pruned by any of the dominance checks during the search and since the algorithm explores all possible actions for each extracted state, we can prove that the plan is found. It first expands the state s_0 by action a_0 (among others) leading to state s_1. Since state s_1 is Pareto-optimal, it cannot be pruned from the priority queue; and therefore, will be expanded eventually. The expansion by action a_1 leads to state s_2 that again cannot be pruned from the priority queue. This procedure is repeated until s_k is extracted from the queue. □

Theorem 6. *The uninformed algorithm ends in a finite number of steps.*

Proof. Since the road graph is finite and there are no negative cycles[7], none of the dominance attributes (time, cost, SoC) can improve infinitely; therefore, eventually all states generated by *expand* function will reach the destination or be pruned by the closed/settled states in corresponding Pareto-set S_v^{cl} and/or solution states Π. □

[6]Charging of greater amount of energy resulting on the same SoC has to be slower

[7]The only possibly negative attribute is the consumption/SoC, and there cannot be a negative consumption cycle due to the law of physics.

Complete Informed Algorithm Properties

In this section, we prove that the algorithm with both heuristics, dimensionality reduction, and contraction hierarchies also finds the whole Pareto-optimal set of travel plans. Since the *uninformed* algorithm finds the whole Pareto-optimal set of travel plans and the heuristics used to extend it to an A*-based algorithm are consistent (see proofs in Sections 8.3.1 and 8.3.2), the algorithm improved with the heuristics also finds the whole Pareto-optimal set of travel plans. The requirements of the dimensionality reduction posed by Pulido et al. [2015] are also satisfied - the lexicographical ordering of the priority queue and heuristic consistency. Contraction hierarchies also do not affect the optimality of the algorithm (Section 8.3.4.2). Therefore, the complete informed algorithm also finds the whole Pareto-optimal set of travel plans.

8.4 Evaluation Setup

In this section, we describe in detail the configurations of the proposed algorithm we evaluated together with used instances of the EV Travel Planning Problem. We also describe evaluation metrics.

8.4.1 Evaluation Problem Instances

The EV planning environments used for the evaluation were constructed from real-world data sets for Germany. Germany has a large road network with many charging stations and good accessibility of data. Besides the large-scale Germany area, we also performed the evaluation on a smaller-scale area of the German state of Bavaria (visualization of the areas can be seen in Figure 8.2). The Bavaria area was used because some of the evaluated algorithm variants could not solve instances for the whole Germany in a reasonable time.

We extracted road *graphs* for both Germany and Bavaria from OpenStreetMaps[8] and then mapped real-world charging stations[9] to them (visualization can be seen in Figure 8.2). The elevation data were gathered from SRTM.[10]

For each region, we created two road graphs - a full graph containing all the roads and a simplified graph without local, residential roads. The usage of the simplified graph (used, e.g., by Schoenberg and Dressler [2023]) is relevant for real-world applications since the local roads are important only for the first and last miles[11] and they can be easily handled

[8] https://download.geofabrik.de/europe/germany.html

[9] https://chargemap.com

[10] https://www2.jpl.nasa.gov/srtm/

[11] Although, there might be shortcuts going through dense residential areas, the gained time is usually very small. Moreover, it is commonly undesirable to guide transiting vehicles via roads meant and designed only for local traffic.

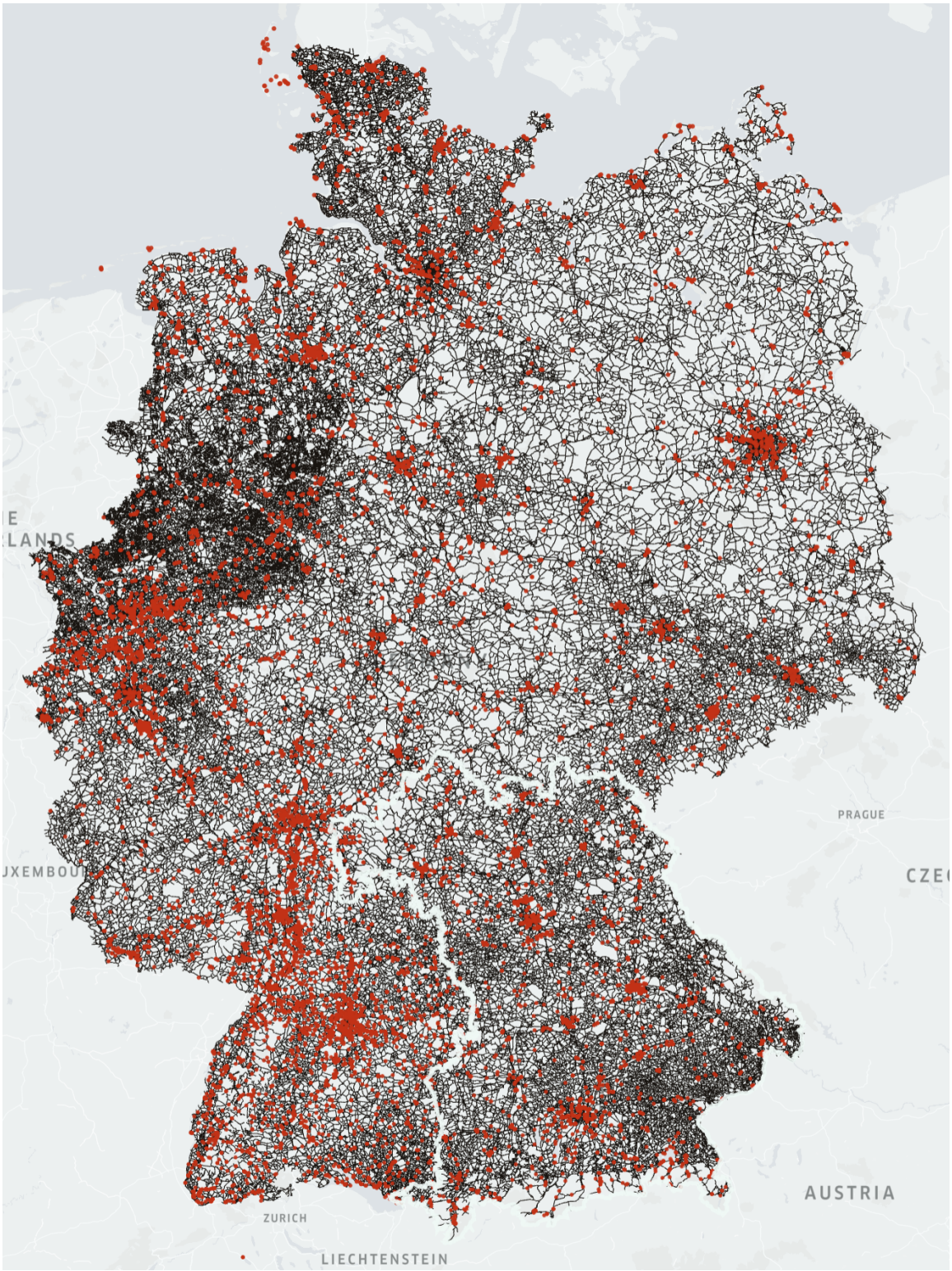


Figure 8.2: Germany road graph with charging stations. Bavaria state outlined bottom right.

by a plan post-processing. However, the contraction hierarchies speed-up performance is significant enough to allow the usage of the full graph.

The EV travel planning environments summary can be seen in the following table.

environment	$\|V\|$	$\|E\|$	$\|Q\|$
Germany	1.5M	3M	12633
Germany-Full	4M	9.2M	12633
Bavaria	300k	600k	2225
Bavaria-Full	800k	1.9M	2225

Table 8.1: The sizes of the EV travel planning environments used in the evaluation: the number of nodes ($|V|$) and the number of edges ($|E|$) of the road network graphs, and the number of charging stations ($|Q|$).

The *charging station* dataset[9] contains 12 633 charging stations (2225 in Bavaria). Each charging station is described by its location (GPS), the maximum power (kW), and the pricing policy. The charging station dataset contains four different types of pricing policies:

- **energy** – The cost is based on the amount of energy the charging station provides, for example, per kilowatt-hour.
- **duration** – The cost is based on charging duration, for example, per minute.
- **fixed** – The user pays a fixed cost no matter how long it takes or how much energy is charged.
- **occupancy** – This policy is very similar to the duration-based policy. The difference is that there is a free period of time during which the occupancy fee is not paid, for example, the first two hours. The reason behind this fee is to motivate users to move out as soon as possible after the EV is charged and to not block the charging station.

The pricing policies can also be combined. For example, the charging cost can depend on the charging duration and also on the amount of energy charged. The resulting charging cost is a sum of the cost produced by both policies. Almost all (99%) of paid charging stations employ the *energy* based policy, the *duration* based policy is used on 27% of paid CS, the *fixed* price on 18%, and the *occupancy* based policy on 26% of paid CS. The pricing policies vary a lot between charging stations, implementing the so-called *location-of-use* pricing.

Since we assume the users will not be willing to wait too long to charge their EVs, we have filtered out charging stations with a maximum charging power of less than 11kW.

We also created a set of charging stations, where all free charging stations are replaced with paid ones to evaluate the impact of free charging stations on the planning time.[12]

[12]For each free charging station, we randomly selected a paid one with the same power and used its pricing policy.

The *charging waiting time* was set to zero $\tau_q(t, \delta^{\mathsf{ch}}) = 0$.

We model the *energy consumption* of the EV with a linear model from Equation 3.2 with the coefficients $\alpha^l = 0.16, \alpha^+ = 1.6$ and $\alpha^- = 1.2$ that with 40kWh *battery capacity* lead to approx. 250km range and with 80kWh battery to 500km range. The model is rather simple; however, the algorithms are consumption model agnostic and can be easily adapted for more complex realistic models with negligible impact on the algorithm performance especially compared to the range influence.

We used a piecewise linear *charging function* similar to Baum et al. [2019b] that expresses well the decreasing charging speed when the state of charge approaches the maximum battery capacity while maintaining simplicity.

To describe the specific charging function, we need to describe *charging efficiency*. It can be written as:

$$\frac{charged\ energy}{power \cdot time}$$

For example, if we want to charge 10kWh on a charging station with 10kW power, the charging would take one hour with 100% efficiency. If the efficiency was 50%, it would take 2 hours.

We model the charging up to 80% of the battery capacity to be 99% efficient, from 80% to 85% to be 86% efficient, from 85% to 90% to be 63% efficient, from 90% to 95% to be 43% efficient and above 95% to be only 15% efficient (see Figure 4.2). It means that it takes more than 6.6× more time to charge the last 5% of the battery than it takes to charge the same amount of energy while the state of charge is below 80%. The parameters are based on the data presented by Zündorf [2014]. Such a non-linear model significantly improves the accuracy of charging time estimates, which is essential for accurate EV travel planning.

We also defined several sets of target charging levels B. We define them by their default step B-step. For example, B-step=10 leads to $\{10\%, 20\%, 30\%, \ldots, 80\%, 85\%, 90\%, 95\%, 100\%\}$ of battery capacity b_{max}. We always include the charging function breakpoints. We use the following B-steps: 1, 5, 10, 20.

The *cost per km* is set to 0, 3, or 10 cents per km.

We generated the planning requests as 1000 random *origin-destination* pairs, uniformly sampled from road graph nodes, for each non-residential graph and then mapped the origin-destination pairs to the full graph (i.e., 4000 in total). The mapping provides a better comparison of the impact of the graph type. The origin-destination pairs were generated so that the direct distance was at least 250km. The distribution of the direct origin-destination distance can be seen in Figure 8.3. The initial SoC was set to 100% of the battery capacity and start time $t_{\mathsf{init}} = 0$.

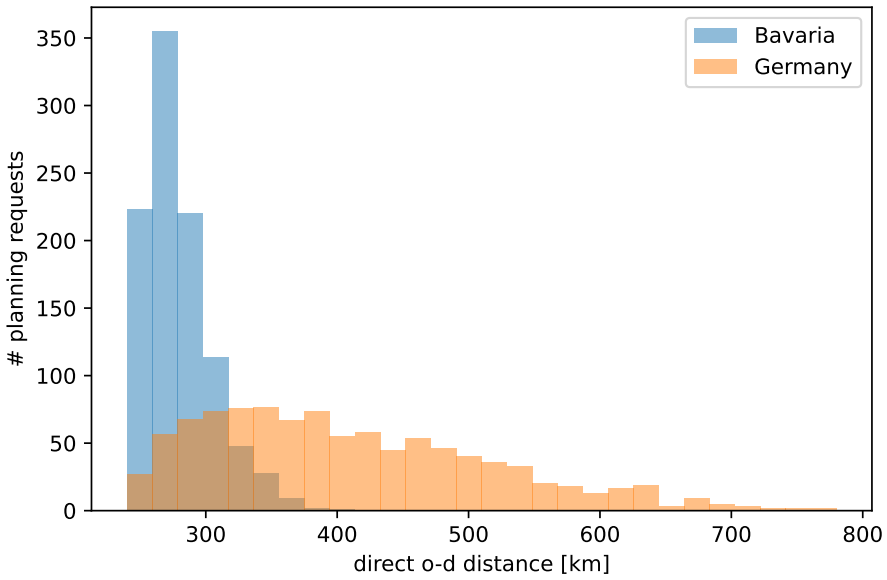


Figure 8.3: The distribution of the direct origin-destination distance for generated travel planning requests.

8.4.2 Evaluated Algorithm Configurations

We first evaluate the impact of the individual components of the *optimal* variant of the algorithm (heuristics and dimensionality reduction) without the CH pre-processing which we evaluate separately. Then, we evaluate the *approximate* variant of the algorithm employing ϵ-relaxation. The evaluation compares the various ϵ-relaxation configurations to the optimal variant in terms of planning time and solution quality (we discuss the evaluation metrics in Section 8.4.3). Specifically, we evaluated each epsilon coefficient—time, cost, and SoC $(\epsilon_t, \epsilon_c, \epsilon_b)$—individually while the remaining coefficients were set to 1.0 and we also evaluated all coefficients combined (all set to the same value). The examined coefficients are: $\{0.995, 0.99, 0.98, 0.96, 0.93, 0.9\}$. Altogether in this paper, we evaluated 24 different ϵ-dominance configurations of the algorithm. Note that our goal in the evaluation was to evaluate the influence of the individual epsilon coefficients on the behavior of the algorithm rather than trying to find a single best possible configuration. In fact, there is not a single best configuration due to the trade-off between the algorithm speed and solution quality and final selection of the coefficients can be considered as fine-tuning that is highly dependent on the specific use case and environment. Finally, we evaluate the impact of the CH pre-processing.

8.4.3 Evaluation Metrics

Since most variants and configurations of the algorithm are not capable of finding the complete solution for more complex problem instances in a reasonable time, we introduced

a 2h time limit after which the planning is terminated. Therefore, we use the *completion percentage* which we define as the percentage of instances for which the algorithm found the complete solution Pareto-set Π within the time limit, as the primary performance metric.

Additionally, we use the *planning time speed-up* as the secondary metric. We define it as t^{base}/t, where t and t^{base} are the *planning times* for the measured algorithm configuration and the baseline configuration, respectively.

Because the ϵ-relaxation technique (Section 3.6) does not preserve optimality, we need to measure also its impact on the *solution quality loss*. We measure the *solution quality loss* as the closeness of the resulting set of EV travel plans to the optimal Pareto-set of plans. From many multi-objective solution quality metrics, surveyed by Laszczyk and Myszkowski [2019] and Riquelme et al. [2015], the best fit for our case is the *average distance* metric proposed by Hrnčíř et al. [2016]:

$$d(\Pi^*, \Pi) = \frac{1}{|\Pi^*|} \sum_{\pi^* \in \Pi^*} \min_{\pi \in \Pi} d(\pi^*, \pi)$$

The average distance of the Pareto-set Π from the full Pareto-set Π^* measures the average Euclidean distance in the objective space (in our case time and money cost) normalized to $[0, 1]$ range. For each objective, the minimum value from all plans $\Pi \cup \Pi^*$ is mapped to 0, and correspondingly, the maximum value is mapped to 1. For illustration, if the optimality loss was 7% equally distributed among the objectives, the distance $d(\pi^*, \pi)$ would be approx. 0.1.

8.4.4 HW and SW Details

We implemented our multi-objective EV travel planning algorithms in Java. We ran the experiments on the OpenJDK 64-Bit Server VM Temurin-17.0.4 JVM on a computing cluster node with 64 cores/128 threads 3.1GHz (2 x AMD EPYC 7543). We ran multiple instances simultaneously while limiting the resources to 8 threads and 31GB of RAM per instance. Due to the high complexity of the problem, we also introduced a time limit of 2 hours.

In summary, we ran experiments for more than 500 combinations of planning environments, EV models, and algorithm configurations. For each combination, solutions for 1000 planning requests were calculated resulting in a total of more than 500 000 problem instances calculated. The CPU time needed to do the computations was more than 2.5 million CPU hours.

8.5 Evaluation Results

In this section, we present the evaluation results of all the speed-ups and heuristics presented in this chapter together with the impact of various problem parameters.

First, we evaluate the impact on planning time of all presented speed-ups that preserve optimality (Section 8.5.1.1) excl. contraction hierarchies which we evaluate separately in Section 8.5.3. We also evaluate the impact on planning time of problem parameters (Section 8.5.1.2). Then, we focus on the impact of the suboptimal ϵ-relaxation technique (Section 3.6) on planning time and solution quality (Section 8.5.2).

Finally, we measure the combined effect (Section 8.5.3) of all proposed speed-ups and heuristics on planning times including the contraction hierarchies pre-processing (Section 8.3.4) and ϵ-relaxation.

8.5.1 Optimal Algorithm without CH Pre-Processing

First, we evaluated the impact of the components of the optimal version of the algorithm, i.e., dimensionality reduction (Section 8.3.3) and the time and cost heuristics (Sections 8.3.1 and 8.3.2). Then, we evaluated the impact of problem parameters on the planning time of the optimal algorithm. The problem and algorithm configuration parameters in the evaluation are the following:

Problem Parameters	
Environment	**Germany**, Bavaria
CS pricing policy	**all**, paid-only
EV range [km]	**250**, 500
ψ [€/km]	0, **0.03**, 0.10
B-step	1, 5, **10**, 20
Algorithm Parameters	
Dimensionality reduction	**on**, off
Time heuristic h_t	**on**, off
Cost heuristic h_c	**on**, off
ϵ_t	**1.0**
ϵ_c	**1.0**
ϵ_b	**1.0**
Contraction hierarchies	**off**

Table 8.2: Overview of used problem and algorithm parameters used for the evaluation of *optimal algorithm*. Default parameters used unless stated otherwise are bold.

8.5.1.1 Evaluating the Impact of Algorithm Parameters

As mentioned above, the optimal algorithm comprises three components (dimensionality reduction and the time and cost heuristics), and each of them can be used independently, i.e., turned on/off. We compared various combinations of the components turned on (e.g., only cost heuristic with dimensionality reduction) to the combination that uses all components, i.e., the full optimal algorithm. Unless stated otherwise, the following planning environment and EV model are used: the whole Germany graph with free charging stations, B-step=10, EV range 250km, and cost per km set to 0.03 €.

Figure 8.4 shows that all individual components are crucial for the optimal algorithm to maintain reasonable performance on country-size graphs (Germany) at least for experimental purposes. Without any of the heuristics, the algorithm would be basically useless, and without dimensionality reduction, the algorithm is capable of solving only 50% of the instances. On smaller planning environments (Bavaria), the algorithm with at least two used components solves a great majority of instances, and even on this much smaller planning environment, the average speed-up of cost heuristic is $\sim 150\times$, of time heuristic $\sim 70\times$ and of the dimensionality reduction $\sim 4\times$.[13] The results also show that dimensionality reduction dramatically reduces the size of stored Pareto-sets and significantly reduces memory requirements since there is a minimum amount of instances terminated because of insufficient memory (purple bars).

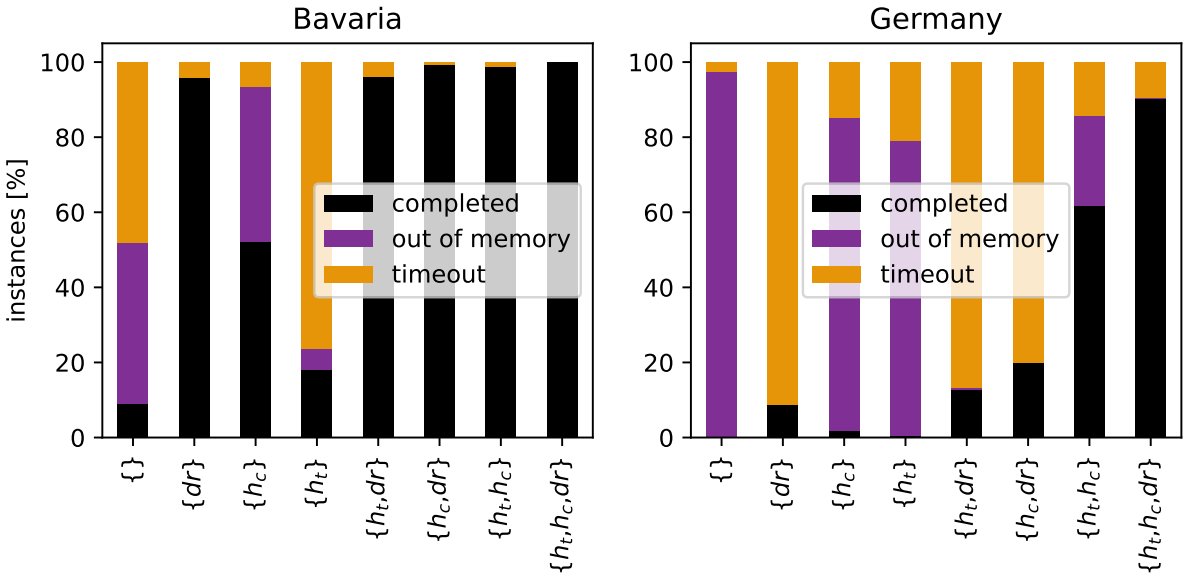


Figure 8.4: Completion percentage for different combinations of optimal algorithm components. h_t – time heuristic, h_c – cost heuristic, dr – dimensionality reduction

[13] All speed-ups are comparisons of the full optimal algorithm without the one specific component. Because of the insufficient performance and too small number of completed instances, we cannot reliably calculate the speed-up on the Germany planning environment.

Although the speed-up of the heuristics and the dimensionality reduction is very significant, the planning time is still not sufficient for solving more complex country-size instances in a real-world application. While the optimal algorithm performs quite well on smaller areas (avg. planning time is 74 seconds on Bavaria), the average planning time on Germany planning environment is more than 18 minutes.

8.5.1.2 Evaluating the Impact of Problem Parameters

We also evaluated the impact of the problem parameters on the planning time of the optimal algorithm. We examined the influence of the availability of free charging, the cost per km of driving, the EV range, and the origin-destination distance.

Contrary to our expectations, free charging stations reduce the planning time. Since the lower bound of the charging part of the cost heuristic is always zero with free charging stations and therefore less informed, we expected it would be a significant performance hit. While the optimal algorithm is capable of successfully solving 92.6% of the planning requests with free charging stations, only paid charging stations reduce the percentage to 33%. The average speed-up with free charging stations is $\sim 11\times$ on completed instances caused by a significant reduction of the average Pareto-set size from 500 with paid-only charging stations to 300 with free charging stations.

The introduction of cost per km of driving leads to increased complexity. As you can see in Table 8.3, zero driving costs allow the algorithm to solve nearly all instances, while the inclusion of cost per km reduces the completion percentage to 83%.

ψ [€/km]	*completion* [%]
0	98.3
0.03	82.6
0.10	82.7

Table 8.3: The number of completed instances for different costs per km (ψ)

The root cause is the dramatically increased sizes of solution Pareto-sets. Figure 8.5 shows that the average Pareto-set size is increased more than 16 times from 33 to 553 for 0.03€/km. It leads to $\sim 2.5\times$ increased average planning time from 480s to 1105s for 0.03€/km. Such a large solution size growth should lead to a much smaller number of completed instances; however, it is largely mitigated by the cost heuristic. The solution size growth is caused by the trade-off of time and distance that is introduced into the cost objective by the cost per km parameter.

[15]The boxplots show median (green line), mean (green triangle), the box showing Q1 (the 25th percentile) and Q3 (the 75th percentile), and the whiskers show the lowest and highest point within 1.5 IQR of the lower and higher quartile respectively. The outliers are shown as circles.

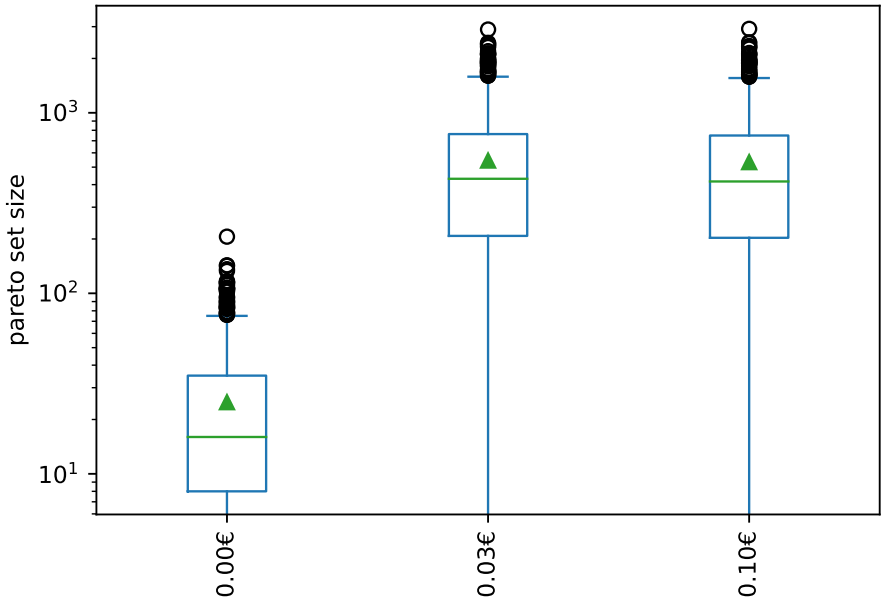


Figure 8.5: Distribution boxplots[15] of the sizes of travel plan Pareto-sets for different costs per km.

Another important parameter of problem instances is the EV range. A greater range leads to a greater completion - 82.6% for 250km range vs. 97.6% for 500km range. More interesting is the great difference in average planning times (1105s vs. 354s) and even greater in median planning times (395s vs. 13s) - see Figure 8.6.

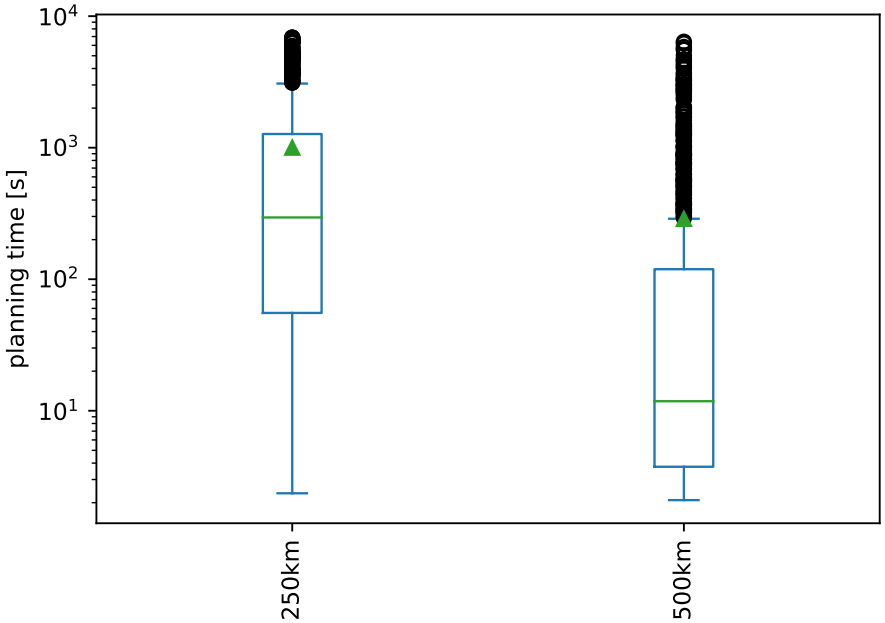


Figure 8.6: Distribution boxplot of planning times for different EV ranges. The green triangles are the means, and the green lines are the medians.

The large planning time drop is caused by a smaller number of required charging

sessions to complete the trip. As you can see in Figure 8.7, over 50% of instances with 500km range EV do not require any charging while some instances with 250km EV range require even more than 2 charging sessions and we can expect that the unsolved scenarios with 250km EV range (the gap to 100% of instances) are also the longer ones. The decreasing planning time with fewer charging sessions is a good promise that with the future extension of EV ranges not only the usability of EVs themselves will improve but also the planning tools will be able to compute better plans on greater distances.

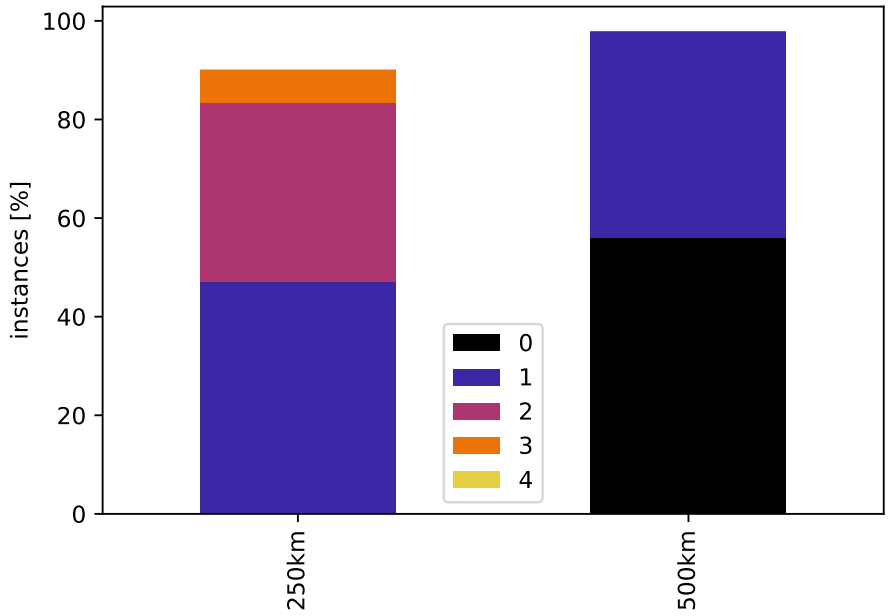


Figure 8.7: Distribution of the minimal number of charging sessions required to get from the request origin to the request destination for different EV ranges.

Figure 8.8 shows the dependency of planning time on the distance between origin and destination. We can see that with shorter distances the correlation is quite high but with increasing distance, the dependency weakens. It appears that around requests requiring at least 2 charging (with 250km range approx. 500km o-d distance) there is another influence, probably the availability of charging infrastructure.

Finally, Figure 8.9 shows the impact of different target charging level discretizations on the planning time and solution quality when compared to very fine discretization with B-step=1. The speed-up is quite significant - on average, the planning time is reduced 8× with B-step=10 (which we use by default) while the solution quality loss is very small. On average the solution quality loss is only 0.013 and only a few outliers have a quality loss greater than 0.05. If even smaller solution quality loss is required then B-step=5 provides a quality loss of 0.008 but with speed-up reduced to 6×. That said, the discretization does not have a significant impact on the solution quality.

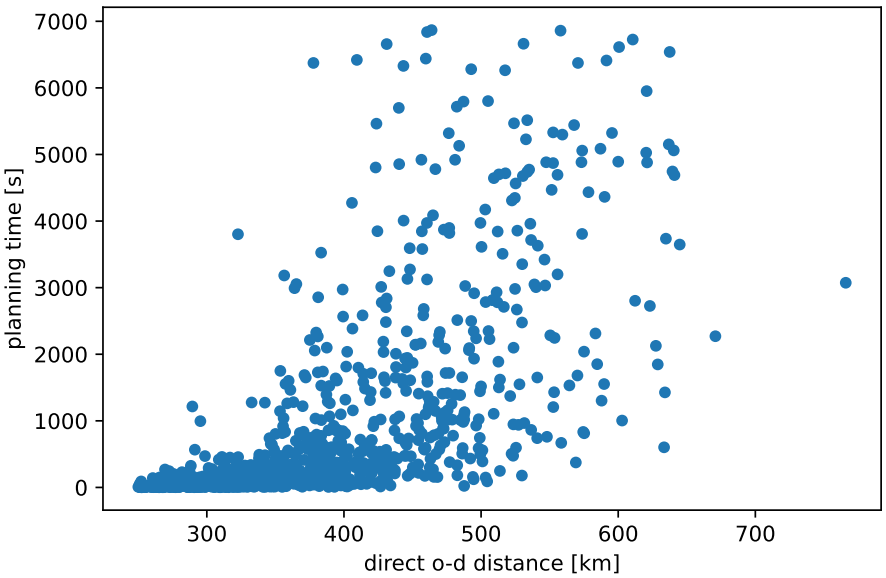


Figure 8.8: Dependency of the planning time on the distance between planning request origin and destination. Each dot represents one planning request from the evaluation request set.

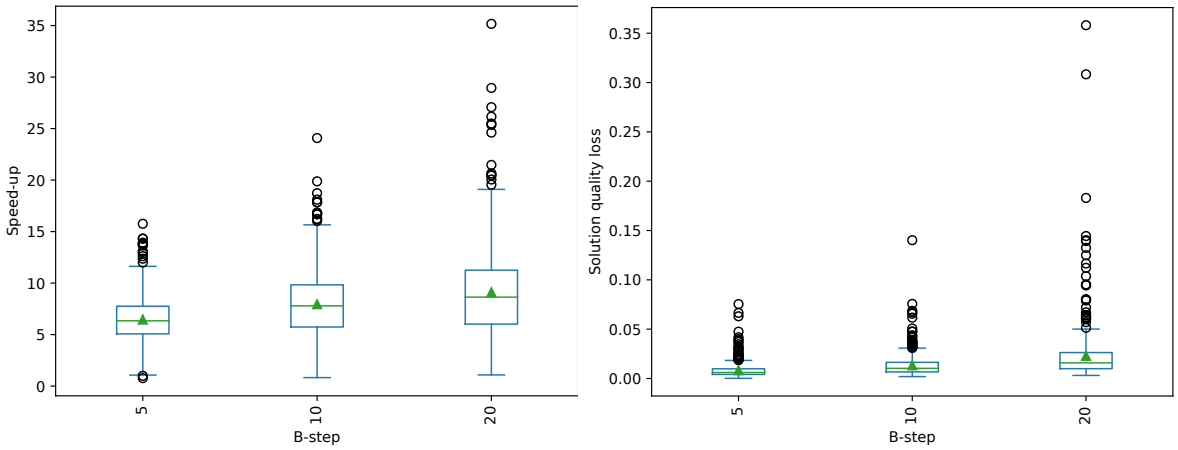


Figure 8.9: Distribution boxplots of speed-ups and solution quality loss obtained by different target charging level discretizations. Baseline - B-step=1.

8.5.2 Approximate Algorithm without CH Pre-Processing

To achieve faster planning times, we need to resort to an approximate algorithm that uses a relaxed dominance relation as described in Section 3.6. The ϵ-relaxation does not preserve optimality; therefore, we need to measure also the impact on the quality of produced travel plans, as described in Section 8.4.3. We ran the experiments on Germany planning environment with 24 combinations of ϵ coefficients described in Section 8.4.2.

The fastest optimal algorithm is used as the baseline – which means that dimensionality reduction and time and cost heuristics are used. An overview of the algorithm and problem parameters can be seen in Table 8.4.

Problem Parameters	
Environment	**Germany**, Bavaria
CS pricing policy	**all**
EV range [km]	**250**
ψ [€/km]	**0.03**
B-step	**10**
Algorithm Parameters	
Dimensionality reduction	**on**
Time heuristic h_t	**on**
Cost heuristic h_c	**on**
ϵ_t	**1.0**, 0.995, 0.99, 0.98, 0.96, 0.93, 0.9
ϵ_c	**1.0**, 0.995, 0.99, 0.98, 0.96, 0.93, 0.9
ϵ_b	**1.0**, 0.995, 0.99, 0.98, 0.96, 0.93, 0.9
Contraction hierarchies	**off**

Table 8.4: Overview of used problem and algorithm parameters used for the evaluation of *approximate algorithm*. Default parameters used unless stated otherwise are bold.

Figure 8.10 shows the trade-off between the achieved speed-up and the loss of solution quality. Quite unexpected is the essentially zero impact of the time component relaxation (the red cluster in the bottom left corner) on both planning time and solution quality. It is probably caused by the lesser impact of the relaxation due to the fact that the dimensionality reduction causes many dominance checks to ignore the time component.

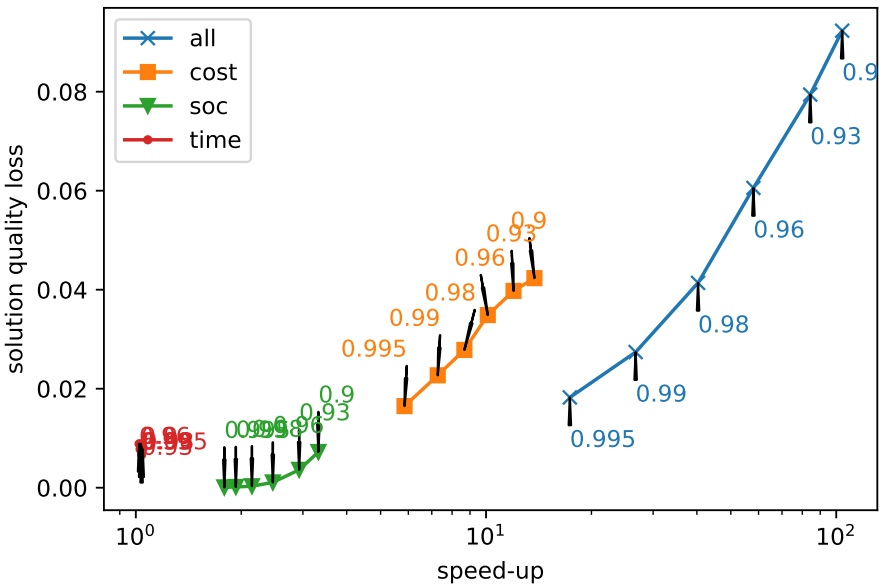


Figure 8.10: The trade-off between average solution quality loss and average speed-up achieved by ϵ-relaxation. The ϵ coefficients are displayed directly in the plot. Lower quality loss and greater speed-up are better.

A very good speed-up of approx. 50× can be achieved while maintaining a very

reasonable solution quality loss of 0.05. In fact, a speed-up of more than 110× while getting 12 travel plans on average can be achieved with the quality loss $\sim$ 0.08. To translate the speed-up to planning time, see Figure 8.11. For practically usable average planning times below 12s (median 6s) on the whole Germany area, we can use the ϵ coefficients set to 0.9. The average Pareto-set size is 12 which provides a good variety of travel plans and the maximum planning time is 126 seconds which is very promising. On the simpler Bavaria planning environment, the average planning time is 4 seconds (median 2s, max 52s) even for the coefficients set to 0.995.

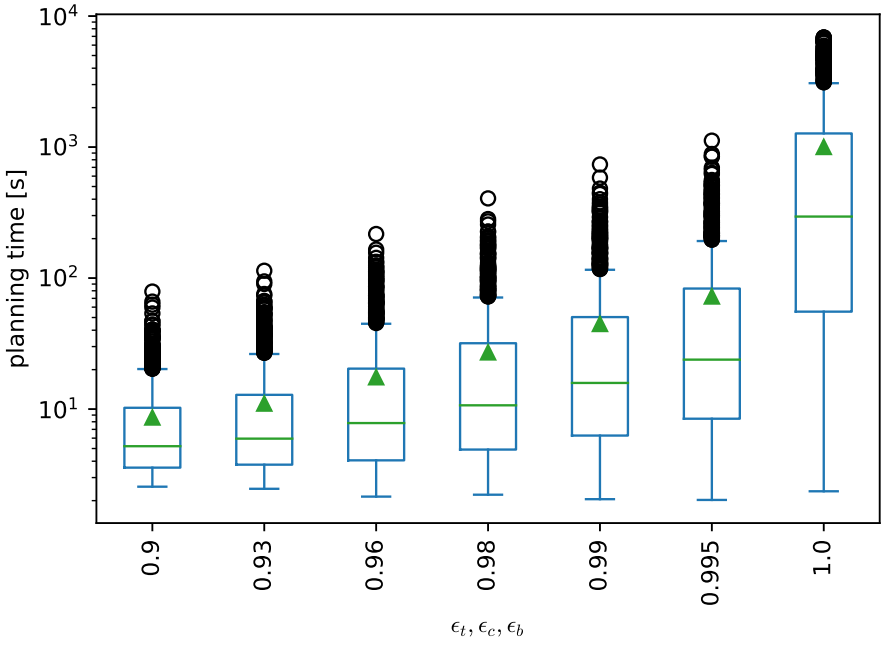


Figure 8.11: Planning time distribution of ϵ-relaxation with *all* coefficients set to the same value.

For a clearer image of how the relaxation impacts the solution quality, you can see Figure 8.12 displaying the optimal solution Pareto-set compared to the Pareto-set affected by the ϵ-relaxation (how these relaxed solutions look on a map can be seen in Figure **??**). It dramatically reduces the number of found travel plans (from 41 to 7), but it still maintains a very good distribution. We can see that the individual plans in the relaxed Pareto-set are distributed among the clusters in the optimal Pareto-set as well as on the map.

Similarly to the informed optimal algorithm, we also evaluated the impact of problem parameters on the planning times of the approximate version of the algorithm. The results were very similar and we, therefore, do not present them in more detail here.

Although our algorithm is capable of solving the EV travel planning problem with time-dependent travel times and also with time-dependent charging pricing policies, we did not perform the evaluation due to the lack of relevant real-world data. However, we

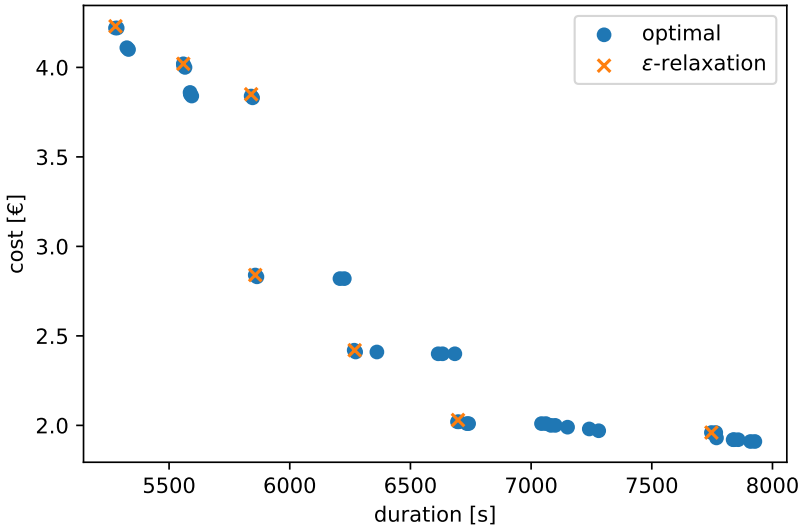


Figure 8.12: Comparison of the optimal Pareto-set to the approximate solution Pareto-set calculated by the algorithm for one request while using ϵ-relaxation with all ϵ coefficients set to 0.95.

estimate that time-dependent travel times would have a negligible impact on the planning time since there would not be a significant change in the number of travel options (the number of roads is still the same). The time-dependent pricing policies would require the introduction of waiting for a cheaper charging price and therefore slightly increasing the number of charging options. However, we estimate the impact on planning time would still be insignificant.

8.5.3 Contraction Hierarchies

We evaluate each phase of the contraction hierarchies speed-up individually. We measure mainly the impact of the parameters of both phases on the planning time.

8.5.3.1 Pre-Processing Phase Evaluation

First, we evaluate the impact of the CH core size (# uncontracted nodes). We evaluate the query performance on the fastest optimal configuration of the algorithm, i.e. with both heuristic and dimensionality reduction, but without $\epsilon-$relaxation.

An overview of the algorithm and problem parameters can be seen in Table 8.7.

In Table 8.6, we can see that CH speeds-up queries significantly. On the smaller Bavaria area where it is capable to solve nearly all instances, the speed-up is more than 6×. On Germany, we need to look first at the number of solved instances. On the non-residential graph, the algorithm without CH solves 82% of the instances (compared to

Problem Parameters	
Environment	Germany {full, non-residential} Bavaria {full, non-residential}
CS pricing policy	**all**
EV range [km]	**250**
ψ [€/km]	**0.03**
B-step	**10**
Algorithm Parameters	
Dimensionality reduction	**on**
Time heuristic h_t	**on**
Cost heuristic h_c	**on**
$\epsilon_t = \epsilon_c = \epsilon_b$	**1.0**
Contraction hierarchies	**on**, off
Core size $\|V^\circ\|$	Bavaria: 3k, 5k, 7k, 10k, 15k Germany: 20k, 30k, 40k, 50k, 75k, 100k

Table 8.5: Overview of used problem and algorithm parameters used for the evaluation of *CH pre-processing phase*. Default parameters used unless stated otherwise are bold.

98% with CH), and on the full graph it solves only 42% of instances. Therefore, direct comparison avg. query time does not have much value. If we compare only the instances that both algorithms can solve, the speed-up on these simpler instances is $\sim 11\times$ on full Germany ($\sim 7\times$ on non-residential Germany). We can assume that more complex instances benefit more from CH.

We can also see that too great or too small core size negatively impacts the performance. The best optimal query performance can be seen around core size of 75k for full Germany and 40k for non-residential Germany (7k and 5k on Bavaria). Although the average query time on full Germany with core size 75k is slightly slower than the rest, it is capable of solving more instances within the time limit. The additional solved instances, which are the more complex ones, probably cause the greater average query time. Besides longer pre-processing time, it appears that too small core also leads to a dramatically increased number of created shortcuts that slow down the query algorithm by exploring too many unnecessary shortcuts.

The main reason why the optimal queries are much slower on the more dense full road graphs is the dramatically increased number of Pareto-optimal solution plans. In the case of Germany, the average Pareto-set size increases from 700 to 1400 (300 to 800 on Bavaria).

		$\lvert V^\circ \rvert$	t_{pr} [min]	$\lvert E^{\mathsf{CH}} \rvert$	# solved	t_{avg} [s]
Bavaria	Non-residential	3k	11.3	746k	1000	11.1
		5k	2.3	651k	1000	9.6
		7k	1.4	615k	1000	10.3
		10k	1.0	583k	1000	11.1
		15k	0.8	547k	1000	13.5
		301k	-	0	1000	73.9
	Full	3k	383.6	2.2M	997	136.5
		5k	33.5	1.9M	999	113.6
		7k	13.0	1.8M	1000	118.8
		10k	7.0	1.7M	999	115.9
		15k	4.5	1.6M	999	126.9
		811k	-	0	964	700.9
Germany	Non-residential	20k	52.8	3.6M	980	570.6
		30k	20.5	3.3M	981	542.4
		40k	10.9	3.1M	982	549.0
		50k	9.6	3.0M	980	554.6
		75k	7.8	2.8M	972	565.2
		100k	7.0	2.7M	971	622.9
		1.5M	-	0	826	1105.1
	Full	20k	1216.6	10.5M	745	1242.6
		30k	224.0	9.4M	772	1258.9
		40k	92.2	8.9M	776	1202.8
		50k	68.3	8.6M	774	1164.4
		75k	46.6	8.2M	782	1221.5
		100k	42.6	7.9M	766	1261.8
		4.1M	-	0	422	1976.5

Table 8.6: Experiment results of the pre-processing phase. $\lvert V^\circ \rvert$ - core size, t_{pr} - pre-processing time, $\lvert E^{\mathsf{CH}} \rvert$ - number of created shortcuts, # solved - number of successfully solved queries, t_{avg} - average query planning time

8.5.3.2 Query Phase Evaluation

We also evaluate the impact of the query phase configurations. We have tried various values of ϵ coefficients. For simplicity, we present here only the configurations where all ϵ coefficients are set to the same value ($\epsilon_t = \epsilon_c = \epsilon_b$).

An overview of the algorithm and problem parameters can be seen in Table 8.7.

For each planning environment, we first tried the CH pre-processing, which appears to have the best performance with the optimal query algorithm. However, the CH best for optimal planning are not always the best with ϵ-relaxation. For example, on full Germany, the best optimal CH with core size 75k has an average query time of 9.4s with $\epsilon = 0.9$, while CH with core size 50k avg. query time is 8.6 (the difference in max. times is greater - 25.1s vs. 20.7s).

Therefore, in Table 8.8, you can see the results on the CH with the fastest avg. query times with ϵ-relaxation - 40k for Non-residential Germany, 50k for full Germany, 5k for

Problem Parameters	
Environment	Germany {full, non-residential} Bavaria {full, non-residential}
CS pricing policy	**all**
EV range [km]	**250**
ψ [€/km]	**0.03**
B-step	**10**
Algorithm Parameters	
Dimensionality reduction	**on**
Time heuristic h_t	**on**
Cost heuristic h_c	**on**
$\epsilon_t = \epsilon_c = \epsilon_b$	**1.0**, 0.995, 0.99, 0.98, 0.96, 0.93, 0.9
Contraction hierarchies	**on**
Core size $\|V^\circ\|$	Bavaria non-residential: **5k** Bavaria full: **7k** Germany non-residential: **40k** Germany full: **50k**

Table 8.7: Overview of used problem and algorithm parameters used for the evaluation of *CH query phase*. Default parameters used unless stated otherwise are bold.

non-residential Bavaria and 7k for full Bavaria.

We can see that, surprisingly, the fastest avg. query times on full Germany are not achieved with the most aggressive relaxation with $\epsilon = 0.9$ but with $\epsilon = 0.93$. The avg. query time difference is relatively small (8.2s vs. 8.6s), and the standard deviation for $\epsilon = 0.93$ is 2.5, which, compared to other measured datasets where the deviation is below 2, indicates that this is probably caused by a noise in this dataset. On full Germany, we can reach very good avg. planning times (below 10s), and if we exclude the residential roads (that could be dealt with by post-processing), we can get even to avg. planning times of 2.6s while the maximum is below 10s which is very good for real-world applications.

Because the ϵ-relaxation technique does not preserve optimality, we need to measure also its impact on the *solution quality loss*. More aggressive pruning leads to smaller solution Pareto-sets (avg. 13 plans with $\epsilon = 0.9$). While the sub-optimal Pareto-sets are dramatically smaller the solution quality loss is very reasonable (below 0.1), which means that the diversity of the Pareto-optimal set of plans is covered well by the subset obtained with ϵ-relaxation. We can also see that the inclusion of the residential roads does not significantly increase the diversity of the solution since the average sizes of the relaxed solutions that maintain good diversity are very similar with or without residential roads.

Figure 8.13 illustrates the trade-off between the achieved speed-up and the solution quality loss (and also the surprising behavior described above). We can see that on full Germany a speed-up of $\sim$150$\times$ (on non-residential even $\sim$170$\times$) can be achieved with

		ϵ	t_{avg} [s]	t_{max} [s]	Avg. $\lvert\Pi\rvert$	Avg. err
Bavaria	Non-residential	-	9.6	406.7	294	0.00
		0.995	0.7	3.4	32	0.02
		0.990	0.5	2.2	26	0.02
		0.980	0.5	1.5	21	0.04
		0.960	0.6	1.1	17	0.06
		0.930	0.4	0.7	12	0.08
		0.900	0.4	0.6	10	0.10
	Full	-	118.8	5810.1	796	0.00
		0.995	2.0	9.0	45	0.02
		0.990	1.8	6.0	33	0.03
		0.980	1.7	3.8	25	0.05
		0.960	1.5	2.4	19	0.08
		0.930	1.3	2.5	12	0.11
		0.900	1.3	1.8	10	0.13
Germany	Non-residential	-	549.0	6809.7	713	0.00
		0.995	11.7	219.5	48	0.01
		0.990	7.5	120.7	37	0.02
		0.980	5.3	64.2	29	0.03
		0.960	3.9	28.8	22	0.04
		0.930	3.2	13.2	15	0.06
		0.900	2.6	8.6	13	0.08
	Full	-	1164.4	6710.4	1422	0.00
		0.995	24.1	395.2	62	0.02
		0.990	16.6	231.7	46	0.02
		0.980	13.0	114.7	34	0.04
		0.960	10.2	63.3	25	0.06
		0.930	8.2	26.0	16	0.08
		0.900	8.6	20.7	13	0.10

Table 8.8: Experiment results of the query phase. ϵ - coefficient used for the ϵ-relaxation speed-up, t_{avg} - average query planning time, t_{max} - maximum query planning time, Avg. $\lvert\Pi\rvert$ - average solution Pareto-set size, Avg. err - average quality loss compared to optimal baseline on instances where both algorithms find the solution

only a minor solution quality loss of 0.08.

8.6 Real-World Prototype Application

Besides the implementation of the algorithm itself, we have also implemented a frontend capable of sending planning requests to the backend algorithm and visualizing the resulting travel plans. The application for EV travel planning in Germany can be accessed online[16]. A screenshot can be seen in Figure 8.14.

[16] `http://its.fel.cvut.cz/ev-travel-planner`

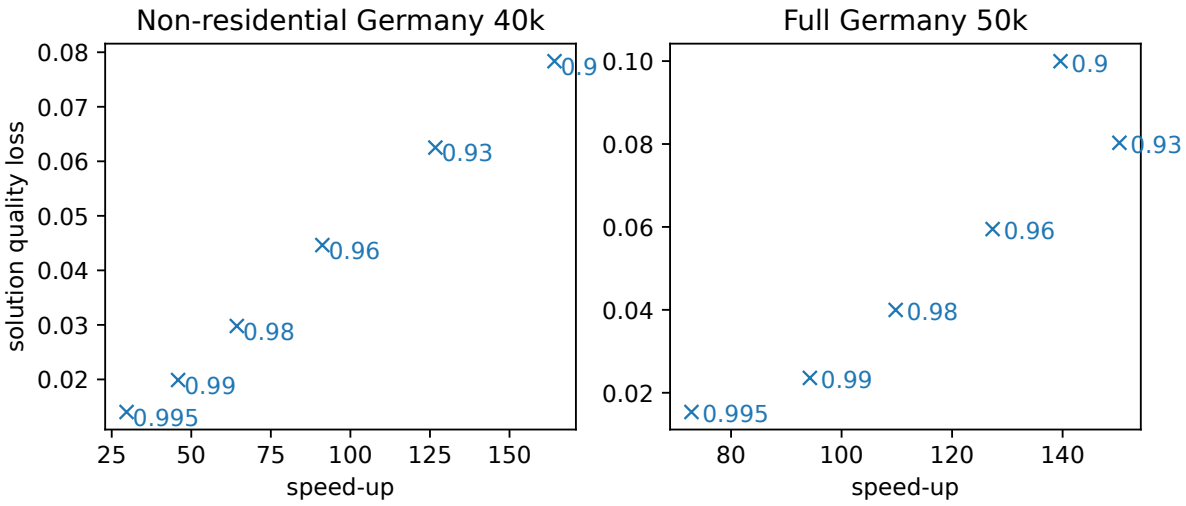


Figure 8.13: The trade-off between average solution quality loss and average speed-up achieved by ϵ-relaxation. The ϵ coefficients are displayed directly in the plot.

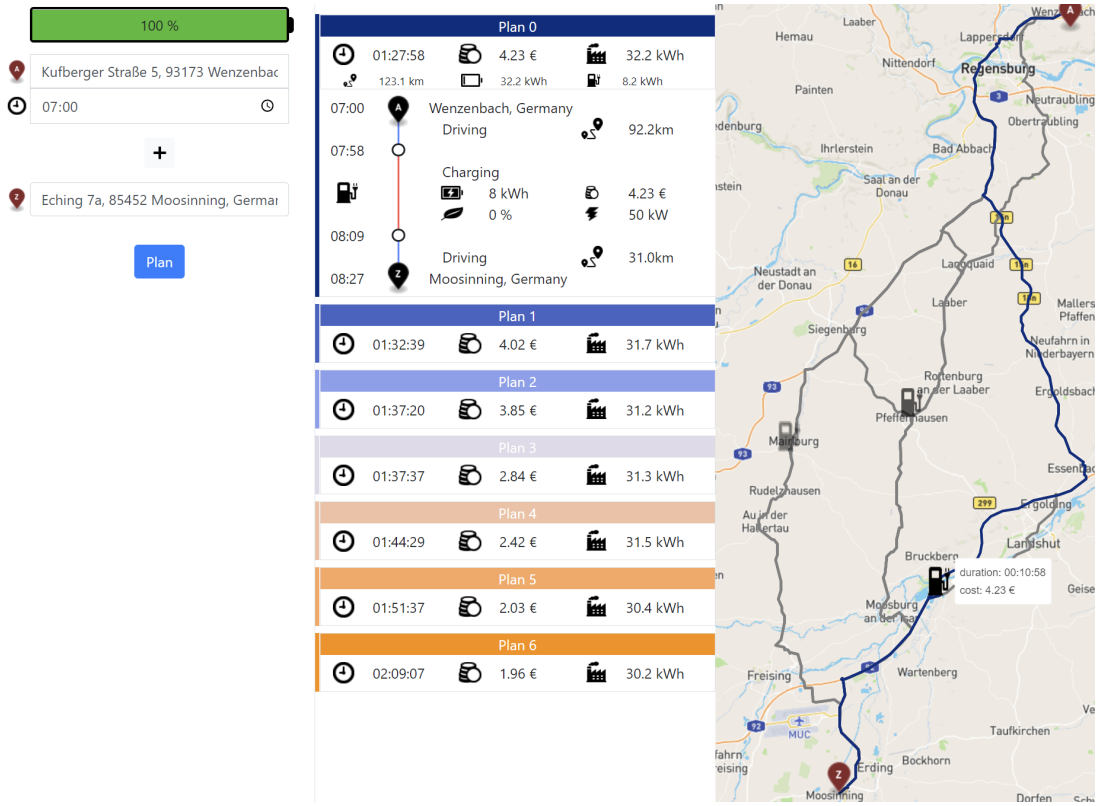


Figure 8.14: Screenshot of the prototype application implementing the multi-objective EV travel planning algorithms described in this chapter. The planning request can be specified in the left panel, the summary of found plans and the detail of a selected plan can be seen in the center, and the right panel displays the plans on the map.

8.7 Summary

In this chapter, we described the multi-objective variant of the EV travel planning problem and proposed an algorithm with a set of speed-up techniques to solve it.

Despite the very high computational complexity of the underlying travel planning problem, we managed to design an approach built on the general algorithm described in Chapter 5 that achieves practically usable planning times with only a minor loss of solution quality. Our proposed algorithm uses several efficient speed-up techniques and can produce EV travel plans, on average, in less than 10 seconds on the whole Germany area (4 seconds without residential roads) and less than 2 seconds on the area of Bavaria (0.5 second without residential roads) with only minor loss of solution quality. The algorithm achieved the planning times by the combination of informed search (heuristics), complexity reduction of the most time-consuming steps of the algorithms (dimensionality reduction), pre-processing (contraction hierarchies) and the sub-optimal pruning of very similar solutions (epsilon-relaxation).

Moreover, we extensively evaluated the impact of key problem and algorithm parameters on the performance of our algorithms and on the quality of produced results, providing useful insights for future improvements. For example, as expected, the algorithms and EV users can benefit from the positive impact of a greater EV range with progress in battery technology. We also showed that the discretization of target charging levels has an insignificant impact on the solution quality while providing an interesting speed-up with the optimal algorithm without CH. Unfortunately, the rising cost of electricity and consequently reduced options of free charging increases the complexity of the problem that is not mitigated enough by the cost heuristic as expected.

Chapter 9

Conclusion

In this thesis, we proposed a novel approach to EV travel planning that takes into account challenges that were not previously considered in this context.

We describe the contributions made while studying and tackling the challenges in Section 9.1 and we also discuss the possible directions of future work in Section 9.2.

9.1 Thesis Contributions

General Problem Definition (Chapter 4) First, we formally defined general multi-objective multi-destination EV travel planning problem that enables us to model a wide variety of concerns to be taken into account when planning EV travel. The proposed problem encompasses all the novel concerns, i.e., multiple destinations, multiple objectives, and incomplete information (details below). Moreover, the proposed problem considers other aspects of EV travel planning that make the problem more realistic, but also more challenging. The problem considers, besides others, a variety of pricing strategies of charging that can be different for each charging station, realistic charging models that take into account decreasing charging speed with increasing battery state of charge and waiting times until charging stations are available. The problem formal definition is necessary for achieving **all goals** of this thesis.

Multi-Destination EV Travel Planning (Chapter 6) Planning EV EV travel plans within the scope of a single trip has its limitations and does not allow, for example, to optimize the travel plan for the entire day when the EV user needs to visit multiple destinations. Therefore, our proposed approach considers multiple temporally constrained destinations.

We designed an algorithm based on the general algorithm described in Chapter 5 together with a time heuristic and two pruning techniques to speed up the search. We

also used the ϵ-dominance relaxation technique and a road graph pre-processing that pre-calculates routes between all charging stations and points of interest.

Our evaluation proved that the holistic multi-destination approach significantly improves the results when compared to the baseline that just chains multiple single-destination plans. The travel plans are improved in terms of plan duration, energy efficiency, and also the overall cost even though the cost is not an optimization objective which shows that the proposed algorithm utilizes the extended optimization scope to find better solutions which is our **Goal 1**. Additionally, we evaluated the impact of the individual speed-up techniques and showed that they are crucial for the scalability of the algorithm.

EV Travel Planning with Incomplete Information (Chapter 7) In the competitive environment of transportation services, such as charging service providers, the providers may not be willing to share all the data that are necessary for EV travel planning since their business competitors may benefit from it. Therefore, our proposed approach considers the incomplete information about the charging infrastructure. More specifically, our approach considers that information about waiting time at charging stations is not available at the start of the planning process and has to be obtained during the planning process by very narrow queries to the service provider API. Since the queries can be time-consuming or there can be a limit on the number of queries, our approach focuses on the minimization of the number of queries.

To solve the problem, we designed an optimal algorithm utilizing two inter-changing phases. The first phase, which is built on the general A* algorithm (Chapter 5), searches for the optimal travel plan based on the already available information. In the second phase, the algorithm selects the information to be obtained and propagates it through the search space.

We evaluated the properties of the problem and their influence on the number of expensive queries on real-world country-scale scenarios. To generate the waiting times, we first generated time intervals (reservations) when the charging stations were occupied. Then, we calculated the waiting times based on the gaps between the reserved time slots.

We showed that the complexity (measured as the number of queries) of the problem grows most likely exponentially with all considered parameters. We also showed that the information propagation via inference that was not used by previous approaches to similar problems can significantly reduce the number of queries.

This contribution fulfills **Goal 2** of the thesis.

Multi-Objective EV Travel Planning (Chapter 8) While the optimization of a single objective was studied extensively, the optimization of multiple objectives in the

context of EV travel planning was studied only to a very limited extent and the few studies that focused on it, proposed algorithms that do not scale to the real-world road networks and realistic number of charging stations. Therefore, we proposed a genuinely multi-objective EV travel planning problem that optimizes both the EV travel plan duration and the overall cost that comprises the cost of charging and the wear-and-tear costs expressed as price per kilometer of driving. The research on the multi-objective problem presented in Chapter 8 is the main contribution of this thesis.

We designed an algorithm that is capable of solving the problem on realistically large country-scale road networks with a realistic number of charging stations while finding the whole Pareto-set of solutions. The algorithm is built on the general multi-objective A* algorithm (Chapter 5) and is enhanced with time and cost heuristics, optimality preserving dimensionality reduction technique [Pulido et al., 2015], multi-objective extension of contraction hierarchies pre-processing [Geisberger et al., 2008], and ϵ-dominance relaxation.

We evaluated the algorithm on real-world country-scale data and showed that the algorithm is capable of solving the problem on road graphs with millions of nodes and more than ten thousand charging stations in seconds with only a minimal loss of the quality of the results. We also extensively evaluated the impact of the individual problem parameters and the individual speed-up techniques to provide a better understanding of the problem and the algorithm which can be very beneficial for future research. This shows that the main **Goal 3** of the thesis was achieved.

Besides the empirical experimental results, we also provided a theoretical discussion on the complexity of the multi-objective problem. We also proved that the proposed algorithm designed to solve it is optimal if not using the ϵ-dominance relaxation technique.

Moreover, we implemented a web-based application that demonstrates the capabilities of the algorithm and can be used as a proof of concept for the real-world use of the algorithm.

9.2 Future Work

In this thesis, we proposed a novel EV travel planning that takes into account three previously unconsidered challenges. However, we focused on the three challenges separately and there is a lot of space for future work that would combine them (as they are defined in the general problem in Chapter 4) and solve them together.

The pre-processing technique that we used for the multi-destination problem (Section 6.2.4) limits the destinations to a predefined set of points of interest. Even though this limitation can be minimized by using, for example, first-/last-mile post-processing that finds the route to the nearest point of interest for each destination, it is worth ex-

ploring the possibility of removing the limit completely by using a more sophisticated pre-processing technique such as multi-objective contraction hierarchies used for solving of the multi-objective problem (Section 8.3.4). The impact of dimensionality reduction on the multi-destination problem planning times is also worth exploring.

To reduce the complexity of the problem, we introduced the discretization of the charge action (Section 4.2). Instead of considering a theoretically infinite number of charging amounts, we only consider a small set of predefined target charging levels when expanding the charge action in the search (for example, 10%, 20%,. . . ,80%, 85%, 90%, 95%, 100% of the battery capacity). Baum et al. [2019b] proposed an approach that avoids this discretization and is able to find the exact amount of energy to be charged without having to generate a very high number of possible charging levels. However, it is designed only for the single-objective problem and its extension to the multi-objective problem, if possible, could provide interesting results. Even though we showed that the impact of the discretization is not significant, it is worth exploring the possibility of removing the discretization completely.

The problem as defined in Chapter 4 does not consider time-dependent travel times and charging prices. The time-dependent charging prices are omitted only for the sake of simplicity (and because of lack of data) and can be easily added to the problem. Even the algorithm can be easily adapted by introducing a waiting time for a cheaper price. Unfortunately, this is not the case for time-dependent travel times. Since we use road graph pre-processing, the time-dependent travel times would have to be handled also in the pre-processing phase. Also, frequent updates of the travel times, such as real-time traffic data, are not directly possible with the used contraction hierarchies pre-processing technique (Section 8.3.4). However, there are variants of contraction hierarchies that can handle both time-dependent travel times [Batz et al., 2009] and real-time updates [Geisberger et al., 2012].

In Chapter 8, we studied the problem that optimizes the travel plan duration and the overall cost. Another possible research direction could be other optimization objectives such as the minimization of the environmental impact (for example, usage of renewable energy) or battery health.

Appendix A

Publications

A.1 Publications Related to the Thesis

A.1.1 Journal Publications

Cuchý et al. [2024a] Marek Cuchý, Michal Jakob, and Jan Mrkos. Route and Charging Planning for Electric Vehicles: A Multi-Objective Approach. *Transportation Letters — Accepted and waiting for publication*, 2024a. ISSN 1942-7867. doi: 10.1080/19427867.2024.2315359. URL `https://doi.org/10.1080/19427867.2024.2315359`

- *Marek Cuchý* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Michal Jakob* performed: conceptualization, methodology, writing - review and editing, supervision
- *Jan Mrkos* performed: writing - review and editing

A.1.2 Other WoS Publications

Cuchý et al. [2024b] Marek Cuchý, Jiří Vokřínek, and Michal Jakob. Multi-objective electric vehicle route and charging planning with contraction hierarchies. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS) — Accepted and waiting for publication*, 2024b

- *Marek Cuchý* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Jiří Vokřínek* performed: writing - review and editing, supervision
- *Michal Jakob* performed: conceptualization, methodology, supervision

Cuchý and Jakob [2019] Marek Cuchý and Michal Jakob. Electric vehicle travel planning with lazy evaluation of recharging times. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3168–3173. IEEE, 2019

- *Marek Cuchý* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Michal Jakob* performed: conceptualization, methodology, writing - review and editing, supervision, funding acquisition
- **2 citations**

Cuchý et al. [2018c] Marek Cuchý, Michal Štolba, and Michal Jakob. Integrated route, charging and activity planning for whole day mobility with electric vehicles. *Lecture Notes in Artificial Intelligence*, 2018c

- *Marek Cuchý* performed: conceptualization, methodology, software, investigation, data curation, writing - review and editing, visualization
- *Michal Štolba* performed: formal analysis, writing - original draft, supervision
- *Michal Jakob* performed: conceptualization, methodology, supervision, funding acquisition
- **3 citations**

A.1.3 Other Publications

Cuchý et al. [2018a] Marek Cuchý, Michal Štolba, and Michal Jakob. Whole day mobility planning with electric vehicles. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART)*, 2018a

- *Marek Cuchý* performed: conceptualization, methodology, software, investigation, data curation, writing - original draft, visualization
- *Michal Štolba* performed: writing - review and editing, supervision
- *Michal Jakob* performed: conceptualization, methodology, writing - review and editing, supervision, funding acquisition
- **11 citations**

A.2 Other Publications

A.2.1 Journal Publications

Kurbanov et al. [2023] Temirlan Kurbanov, Marek Cuchý, and Jiří Vokřínek. Fast one-to-many multicriteria shortest path search. *IEEE Transactions on Intelligent Transportation Systems*, 2023

- *Temirlan Kurbanov* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Marek Cuchý* performed: conceptualization, methodology, data curation
- *Jiří Vokřínek* performed: writing - review and editing, supervision
- **2 citations**

A.2.2 Other WoS Publications

Cuchý et al. [2018b] Marek Cuchý, Michal Štolba, and Michal Jakob. Benefits of multi-destination travel planning for electric vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 327–332. IEEE, 2018b

- *Marek Cuchý* performed: conceptualization, methodology, software, investigation, data curation, writing - original draft, visualization

- *Michal Štolba* performed: writing - review and editing, supervision
- *Michal Jakob* performed: conceptualization, methodology, writing - review and editing, supervision
- **6 citations**

Kurbanov et al. [2022] Temirlan Kurbanov, Marek Cuchý, and Jiří Vokřínek. Heuristics for fast one-to-many multicriteria shortest path search. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 594–599. IEEE, 2022

- *Temirlan Kurbanov* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Marek Cuchý* performed: conceptualization, methodology, data curation
- *Jiří Vokřínek* performed: writing - review and editing, supervision

Basmadjian et al. [2019] Robert Basmadjian, Benedikt Kirpes, Jan Mrkos, Marek Cuchý, and Sahar Rastegar. An Interoperable Reservation System for Public Electric Vehicle Charging Stations: A Case Study in Germany. In *Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities*, TESCA'19, pages 22–29. Association for Computing Machinery, 2019. ISBN 978-1-4503-7015-8. doi: 10.1145/3364544.3364825. URL `https://doi.org/10.1145/3364544.3364825`

- *Basmadjan Robert* performed: conceptualization, methodology, writing - original draft, writing - review and editing, funding acquisition
- *Benedikt Kirpes* performed: conceptualization, methodology, data curation, writing - original draft, writing - review and editing, visualization
- *Jan Mrkos* performed: methodology, software, formal analysis, investigation, data curation, writing - original draft, writing - review and editing, visualization
- *Marek Cuchý* performed: methodology, software, investigation, data curation
- **5 citations**

Čertický et al. [2015] Michal Čertický, Jan Drchal, Marek Cuchý, and Michal Jakob. Fully agent-based simulation model of multimodal mobility in european cities. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 229–236. IEEE, 2015

- *Michal Čertický* performed: conceptualization, methodology, software, formal analysis, investigation, data curation, writing - original draft, visualization
- *Jan Drchal* performed: writing - original draft, conceptualization, methodology, data curation
- *Marek Cuchý* performed: methodology, software, data curation, visualisation
- *Michal Jakob* performed: writing - review and editing, supervision
- **23 citations**

Bibliography

Y. P. Aneja and K. P. K. Nair. The constrained shortest path problem. *Naval Research Logistics Quarterly*, 25(3):549–555, 1978. ISSN 1931-9193. doi: 10.1002/nav.3800250314. URL `http://dx.doi.org/10.1002/nav.3800250314`.

Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. In *Annual Conference on Artificial Intelligence*, pages 309–316. Springer, 2010.

Robert Basmadjian, Benedikt Kirpes, Jan Mrkos, Marek Cuchý, and Sahar Rastegar. An Interoperable Reservation System for Public Electric Vehicle Charging Stations: A Case Study in Germany. In *Proceedings of the 1st ACM International Workshop on Technology Enablers and Innovative Applications for Smart Cities and Communities*, TESCA'19, pages 22–29. Association for Computing Machinery, 2019. ISBN 978-1-4503-7015-8. doi: 10.1145/3364544.3364825. URL `https://doi.org/10.1145/3364544.3364825`.

Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

Lucas S Batista, Felipe Campelo, Frederico G Guimarães, and Jaime A Ramírez. A comparison of dominance criteria in many-objective optimization problems. In *Evolutionary Computation (CEC), 2011 IEEE Congress on*, pages 2359–2366. IEEE, 2011.

G Veit Batz, Daniel Delling, Peter Sanders, and Christian Vetter. Time-dependent contraction hierarchies. In *2009 Proceedings of the Eleventh Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 97–105. SIAM, 2009.

Reinhard Bauer and Daniel Delling. Sharc: Fast and robust unidirectional routing. *Journal of Experimental Algorithmics (JEA)*, 14:2–4, 2009.

Moritz Baum. *Engineering route planning algorithms for battery electric vehicles*. PhD thesis, Karlsruher Instituts für Technologie, 2018.

Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, 2019a.

Moritz Baum, Julian Dibbelt, Thomas Pajor, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Energy-optimal routes for battery electric vehicles. *Algorithmica*, pages 1–57, 2019b.

Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, 16(1):87–90, 1958.

Sirine Ben Abbes, Lilia Rejeb, and Lasaad Baati. Route planning for electric vehicles. *IET Intelligent Transport Systems*, 2022.

Hans-Joachim Bockenhauer, Juraj Hromkovic, Joachim Kneis, and Joachim Kupke. The parameterized approximability of tsp with deadlines. *Theory of Computing Systems*, 41(3):431–444, Oct 2007. ISSN 1433-0490. doi: 10.1007/s00224-007-1347-x. URL `https://doi.org/10.1007/s00224-007-1347-x`.

Blai Bonet and Hector Geffner. Planning with incomplete information as heuristic search in belief space. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems*, pages 52–61. AAAI Press, 2000.

Michal Čertický, Jan Drchal, Marek Cuchý, and Michal Jakob. Fully agent-based simulation model of multimodal mobility in european cities. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 229–236. IEEE, 2015.

Benjamin Cohen, Mike Phillips, and Maxim Likhachev. Planning single-arm manipulations with n-arm robots. In *Proceedings of Robotics: Science and Systems*, 2015.

Gérard Cornuéjols, Jean Fonlupt, and Denis Naddef. The traveling salesman problem on a graph and some related integer polyhedra. *Mathematical Programming*, 33(1):1–27, Sep 1985. ISSN 1436-4646. doi: 10.1007/BF01582008. URL `https://doi.org/10.1007/BF01582008`.

Marek Cuchý and Michal Jakob. Electric vehicle travel planning with lazy evaluation of recharging times. In *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, pages 3168–3173. IEEE, 2019.

Marek Cuchý, Michal Štolba, and Michal Jakob. Whole day mobility planning with electric vehicles. In *Proceedings of the 10th International Conference on Agents and Artificial Intelligence (ICAART)*, 2018a.

Marek Cuchý, Michal Štolba, and Michal Jakob. Benefits of multi-destination travel planning for electric vehicles. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 327–332. IEEE, 2018b.

Marek Cuchý, Michal Štolba, and Michal Jakob. Integrated route, charging and activity planning for whole day mobility with electric vehicles. *Lecture Notes in Artificial Intelligence*, 2018c.

Marek Cuchý, Michal Jakob, and Jan Mrkos. Route and Charging Planning for Electric Vehicles: A Multi-Objective Approach. *Transportation Letters — Accepted and waiting for publication*, 2024a. ISSN 1942-7867. doi: 10.1080/19427867.2024.2315359. URL `https://doi.org/10.1080/19427867.2024.2315359`.

Marek Cuchý, Jiří Vokřínek, and Michal Jakob. Multi-objective electric vehicle route and charging planning with contraction hierarchies. In *Proceedings of the 34th International Conference on Automated Planning and Scheduling (ICAPS) — Accepted and waiting for publication*, 2024b.

Christopher M Dellin and Siddhartha S Srinivasa. A unifying formalism for shortest path problems with expensive edge evaluations via lazy best-first search over paths with edge selectors. In *Proceedings of the 26th International Conference on Automated Planning and Scheduling (ICAPS)*, pages 459–467, 2016.

Daniel Delling and Dorothea Wagner. Pareto paths with sharc. In *International Symposium on Experimental Algorithms*, pages 125–136. Springer, 2009.

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

Jochen Eisner, Stefan Funke, and Sabine Storandt. Optimal route planning for electric vehicles in large networks. In *Proceedings of the 25th Conference on Artificial Intelligence*, pages 1108–1113, 2011.

David Eppstein and Michael T Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–10, 2008.

Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *Experimental Algorithms: 7th International Workshop, WEA 2008 Provincetown, MA, USA, May 30-June 1, 2008 Proceedings 7*, pages 319–333. Springer, 2008.

Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

Pierre Hansen. Bicriterion path problems. In *Multiple criteria decision making theory and application*, pages 109–127. Springer, 1980.

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.

John A Hartigan and Manchek A Wong. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1): 100–108, 1979.

Jan Hrnčíř, Pavol Žilecký, Qing Song, and Michal Jakob. Practical multicriteria urban bicycle routing. *IEEE Transactions on Intelligent Transportation Systems*, 18(3):493–504, 2016.

Donald B Johnson. Efficient algorithms for shortest paths in sparse networks. *Journal of the ACM (JACM)*, 24(1):1–13, 1977.

Temirlan Kurbanov, Marek Cuchý, and Jiří Vokřínek. Heuristics for fast one-to-many multicriteria shortest path search. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 594–599. IEEE, 2022.

Temirlan Kurbanov, Marek Cuchý, and Jiří Vokřínek. Fast one-to-many multicriteria shortest path search. *IEEE Transactions on Intelligent Transportation Systems*, 2023.

Maciej Laszczyk and Paweł B Myszkowski. Survey of quality measures for multi-objective optimization: Construction of complementary set of multi-objective quality measures. *Swarm and Evolutionary Computation*, 48:109–133, 2019.

Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *Evolutionary computation*, 10(3):263–282, 2002.

Yi Liu, Xuesong Feng, Lukai Zhang, Weixing Hua, and Kemeng Li. A pareto artificial fish swarm algorithm for solving a multi-objective electric transit network design problem. *Transportmetrica A: Transport Science*, 16(3):1648–1670, 2020.

Lawrence Mandow, JL Pérez De la Cruz, et al. A new approach to multiobjective A* search. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, volume 8, 2005.

Ernesto Queiros Vieira Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

Rolf H Möhring, Heiko Schilling, Birk Schütz, Dorothea Wagner, and Thomas Willhalm. Partitioning graphs to speedup dijkstra's algorithm. *Journal of Experimental Algorithmics (JEA)*, 11:2–8, 2007.

Matthias Müller-Hannemann and Karsten Weihe. On the cardinality of the pareto set in bicriteria shortest path problems. *Annals of Operations Research*, 147(1):269–286, 2006.

Venkatraman Narayanan and Maxim Likhachev. Heuristic search on graphs with existence priors for expensive-to-evaluate edges. In *Proceedings of the 27th International Conference on Automated Planning and Scheduling (ICAPS)*, 2017.

Evdokia Nikolova and David R Karger. Route planning under uncertainty: The canadian traveller problem. In *Proceedings of the 23rd Conference on Artificial Intelligence*, pages 969–974, 2008.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016. ISBN 978-3-319-28927-4.

Joseph F Pekny and Donald L Miller. An exact parallel algorithm for the resource constrained traveling salesman problem with application to scheduling with an aggregate deadline. In *Proceedings of the 1990 ACM annual conference on Cooperation*, pages 208–214. ACM, 1990.

Mike Phillips, Maxim Likhachev, and Sven Koenig. PA* SE: Parallel A* for slow expansions. In *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS)*, 2014.

Francisco-Javier Pulido, Lawrence Mandow, and José-Luis Pérez-de-la Cruz. Dimensionality reduction in multiobjective shortest path search. *Computers & Operations Research*, 64:60–70, 2015.

Payas Rajan, Moritz Baum, Michael Wegner, Tobias Zündorf, Christian J West, Dennis Schieferdecker, and Daniel Delling. Robustness generalizations of the shortest feasible path problem for electric vehicles. In *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

Michael N. Rice and Vassilis J. Tsotras. Parameterized algorithms for generalized traveling salesman problems in road networks. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL'13, pages 114–123, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2521-9. doi: 10.1145/2525314.2525342. URL `http://doi.acm.org/10.1145/2525314.2525342`.

Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, pages 1–11. IEEE, 2015.

R. Roberti and M. Wen. The electric traveling salesman problem with time windows. *Transportation Research Part E: Logistics and Transportation Review*, 89:32 – 52, 2016. ISSN 1366-5545. doi: https://doi.org/10.1016/j.tre.2016.01.010. URL `http://www.sciencedirect.com/science/article/pii/S1366554516000181`.

Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Proceedings of the 25th Conference on Artificial Intelligence*, 2011.

Peter Sanders and Dominik Schultes. Engineering highway hierarchies. In *ESA*, volume 6, pages 804–816. Springer, 2006.

Michael Schneider, Andreas Stenger, and Dominik Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4): 500–520, 2014.

Sven Schoenberg and Falko Dressler. Reducing waiting times at charging stations with adaptive electric vehicle route planning. *IEEE Transactions on Intelligent Vehicles*, 8 (1):95–107, 2023.

René Schönfelder, Martin Leucker, and Sebastian Walther. Efficient profile routing for electric vehicles. In *International Conference on Internet of Vehicles*, pages 21–30. Springer, 2014.

Umair Farooq Siddiqi, Yoichi Shiraishi, and Sadiq M Sait. Multi-constrained route optimization for electric vehicles (evs) using particle swarm optimization (pso). In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 391–396. IEEE, 2011.

Bradley S Stewart and Chelsea C White III. Multiobjective A*. *Journal of the ACM (JACM)*, 38(4):775–814, 1991.

Sabine Storandt. Quick and energy-efficient routes: computing constrained shortest paths for electric vehicles. In *Proceedings of the 5th ACM SIGSPATIAL international workshop on computational transportation science*, pages 20–25, 2012.

Sabine Storandt and Stefan Funke. Cruising with a battery-powered vehicle and not getting stranded. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Cong Quoc Tran, Mehdi Keyvan-Ekbatani, Dong Ngoduy, and David Watling. Stochasticity and environmental cost inclusion for electric vehicles fast-charging facility deployment. *Transportation Research Part E: Logistics and Transportation Review*, 154: 102460, 2021.

Volkan Ünal, Mehmet Soysal, Mustafa Çimen, and Çağrı Koç. Dynamic routing optimization with electric vehicles under stochastic battery depletion. *Transportation Letters*, pages 1–13, 2022.

Takayuki Yoshizumi, Teruhisa Miura, and Toru Ishida. A* with partial expansion for large branching factor problems. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 923–929, 2000.

Siying Zhu. Multi-objective route planning problem for cycle-tourists. *Transportation Letters*, 14(3):298–306, 2022.

Tobias Zündorf. Electric vehicle routing with realistic recharging models. Master's thesis, Department of Informatics Karlsruhe, Germany, 2014.