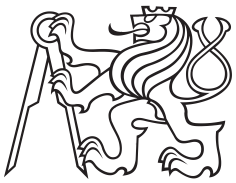
Master Thesis



Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Computer Science

Route Planning for Electric Vehicles with Charging Stops and Points of Interest

Martin Vybíralík

Supervisor: Ing. Antonín Novák
May 2022

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Vybíralík** Jméno: **Martin** Osobní číslo: **474403**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Specializace: **Umělá inteligence**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Plánování tras elektromobilu s nabíjecími zastávkami a body zájmu

Název diplomové práce anglicky:

Route Planning for Electric Vehicles with Charging Stops and Points of Interest

Pokyny pro vypracování:

The goal of the thesis is the development of a scalable algorithm for route planning for battery electric vehicles.
The emphasis should be put on a realistic model of the road network, vehicle consumption, and charging characteristics.
1. Survey existing literature focused on the topic.
2. Define the problem of route planning for battery electric vehicles with charging stops and points of interest.
3. Design an algorithm solving the problem, considering realistic models of vehicle consumption, charging, and road network.
4. Implement the designed algorithm.
5. Evaluate the implemented algorithm on real-world data and compare it with related methods.

Seznam doporučené literatury:

[1] Baum, M., Dibbelt, J., Gemsa, A., Wagner, D., & Zündorf, T. (2019). Shortest feasible paths with charging stops for battery electric vehicles. Transportation Science, 53(6), 1627-1655.
[2] Eisner, J., Funke, S., & Storandt, S. (2011, August). Optimal route planning for electric vehicles in large networks. In Twenty-Fifth AAAI Conference on Artificial Intelligence.
[3] Bast, H., Delling, D., Goldberg, A., Müller-Hannemann, M., Pajor, T., Sanders, P. & Werneck, R. F. (2016). Route planning in transportation networks. In Algorithm engineering (pp. 19-80). Springer, Cham

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Antonín Novák katedra řídicí techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **28.01.2022** Termín odevzdání diplomové práce: **20.05.2022**

Platnost zadání diplomové práce: **30.09.2023**

Ing. Antonín Novák podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Acknowledgements

I would like to express my sincere gratitude to my supervisor Ing. Antonín Novák for his endless enthusiasm and energy spent on our cooperation. I am firmly convinced that my results would be miles and miles far from the current state without his helping hand.

I would like to greatly appreciate the support from my girlfriend Anna Brunclíková. She has always been standing right by my side, providing me with her honest point of view.

Last but not least, my deepest appreciation goes to my family for their infinite patience and warm encouragement whenever it was needed.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 19. May 2022

Abstract

This diploma thesis deals with the problem of route planning and navigation for electric vehicles (EVs) together with the scheduling of charging stops, respecting the feasibility of produced plans (not running out of energy), and user's intentions in visiting different points of interest (POIs). It focuses on the design of planning algorithms and optimizing the overall time of the route while incorporating POIs into the planning process. Significant effort is made to achieve the best possible scalability, ensuring reasonable runtimes on continental-scale regions and enabling deployment in real-world navigation systems. Considerable attention is also paid to elaborating a realistic EV consumption and charging model, which are essential to capture real-world conditions adequately. An approach presented in this thesis follows up on the current state-of-the-art solutions based on Bicriteria Shortest Path algorithms (BSP). Several adaptations and new approaches in data preprocessing and additional constraints modeling (POIs) are introduced with a heuristic technique based on precomputing charging plans within several specific locations. The final solution is evaluated on real-world datasets of charging stations and regions capturing the northern, central and southern Europe region enriched by its elevation profile.

Keywords: electric vehicles, path planning, charging, charging curve, consumption, points of interest, multicriteria optimization

Supervisor: Ing. Antonín Novák

Abstrakt

Tato diplomová práce se zabývá problémem navigace a plánováním tras pro elektrická vozidla společně s plánovaním zastávek na dobíjení. Je kladen důraz na zachování validnosti výsledných plánů (v průběhu cesty nedojde k vyčerpání elektrické energie automobilu) a modelování uživatelem zvolených podmínek na navštívení různých bodů zájmu. Zaměřuje se především na návrh plánovacích algoritmů optimalicujících celkový čas cesty a začlenění bodů zájmu do plánovacího procesu. Značné úsilí je věnováno dosažení co možná nejlepší škálovatelnosti, z níž vyplývají rozumné časy běhu jednotlivých dotazů při použití map kontinentálního měřítka. To vše je motivováno možností nasazení algoritmu do produkčních navigačních systémů. Nemalá pozornost je věnována rovněž použitým modelům spotřeby a nabíjení elektromobilu, které jsou pro zachycení podmínek reálného světa zásadní. Přístup použitý v této práci vychází z řešení prezentovaných v současné literatuře, jež jsou založena na algoritmech pro bikriteriální nejkratší cesty (BSP). Přínosem této práce je především představení nových přístupů v oblasti předzpracování dat, modelování přídavných podmínek vrámci optimalizačního algoritmu a implementace nové heuristické techniky založené na předpočítání nabíjecích plánů mezi několika specifickými místy. Výsledné řešení je vyhodnoceno na reálných datasetech nabíjecích stanic a map zachycujících oblasti severní, střední a jižní Evropy, jež jsou obohaceny o elevační profil.

Klíčová slova: elektrická auta, plánování tras, nabíjení, nabíjecí křivka, spotřeba, body zájmu, multikriteriální optimalizace

Překlad názvu: Plánování tras elektromobilu s nabíjecími zastávkami a body zájmu

Contents

Figures

Tables

Chapter 1

Introduction

1.1 Motivation

In the light of the recent decades, when the question of ecology has become one of the essential topics discussed throughout society. There has been an effort to invent, develop and implement new environment-friendly techniques and methods in almost every possible field of human activity. Through elaborating on questions of pollution, climate change, natural resource depletion, deforestation or generation of unsustainable waste, the world has clung to fields of renewable sources of electric energy, limitation of fossil fuels usage, waste reduction by introducing sustainable packaging solutions, water waste reduction by the usage of environment-friendly appliances and so on.

Omitting electric vehicles (EVs) in any debate dedicated to ecology and innovative environment-friendly solutions would be unthinkable. It is a controversial and awkward subject in various distinct criteria, namely the higher purchase prices, lack of charging stations, not adequately planned and prepared infrastructure of the electric network, the question of safety, or battery degradation. Nevertheless, if no rapid changes in related fields happen, EVs seem to represent a considerable part of the future transportation field. Numerous projects supporting the spread of electromobility throughout the world reinforce this statement. Electric Mobility Europe or UNEP's Global Electric Mobility Programme are just a few chosen projects and organizations with the field of application in Europe dedicated to developing electromobility.

1.2 Electric Vehicles in Transportation

There are several technical characteristics of EVs which substantially affect their usage. Starting from the most known issues such as limited driving range compared to conventional combustion engine cars, the limited network of charging stations, longer charging times, through more complicated subjects like modeling distinct charging curves of individual vehicles dependent on the current SoC up to modeling EV consumption affected by conditions like elevation profile of the path, the weather, driving style, or technical parameters of the vehicle.

Contrarily to assumptions and habits of refueling vehicle drivers originating from almost unlimited access to refueling stations, negligible time requirements on refueling, and large driving ranges, it is considerably harder to determine the most efficient routes or charging plans taking into account criteria such as total driving time, energy consumption, cost, reliability or even requirements for visiting some Points of Interest (POIs) during a planned journey. Systems and algorithms taking into account all of these attributes help drivers of EVs better prepare their routes and contribute to the feasibility of the whole electromobility sector.

Such systems should provide EV drivers with available personifications and possibilities to adapt produced plans to meet most of their requirements. A high-ranking manager has distinct ideas about the perfect route plan for his business trip than family traveling on their vacation. Therefore, there is a natural need to introduce a set of algorithms or techniques producing "no time to lose" fastest plans assuming a dynamic driving style motivated by "time is money" and peaceful conservative, and calm route plans, including food stops or stops for shopping.

Another key feature is the ability to scale in large geographic regions, supported by a claim that benefits resulting from properly planned charging schedules and thoroughly scheduled stops for visiting different kinds of POIs manifest themselves the most considering long-range journeys with several charging stops, where the stress on good timing is substantial.

On the other hand, considering shorter routes with varying SoC at the beginning and desired SoC at the destination, it is essential to focus on modeling EV consumption and charging process to capture the reality as much as possible. Nevertheless, even the best model still produces just an estimation, so features supporting robustness and preventing running out of energy are necessary to plugin easily. When considering any extra stops (food stops, shopping stops, sightseeing) together with scheduling charging for EVs, there is no doubt that it is beneficial to parallelize these activities and focus on planning these extra stops nearby charging locations.

1.3 Related Work

1.3.1 Shortest Path Problem (SP)

Dijkstra's algorithm [13] is probably the most known algorithm in this field. However, it does not scale enough to satisfy the requirements of current routing applications working with large-scale areas or online data with an extreme focus on the algorithm runtime. For this reason, numerous adaptations and even completely distinct approaches to solving the SP problem have been developed.

A survey paper [4] written by Hannah Bast et al. serves as an overview of currently known and widely used approaches for the SP problem introducing basic Dijkstra's [13], Bellman-Ford [10], Floyd-Warshall [20] algorithms, goal-directed techniques such A^* [26], ALT [23] or Arc flags [28, 34], hierar-

chical techniques with its probably most known representative Contraction Hierarchies [22] and bounded-hop techniques based on precomputing distances between pairs of vertices and using these values as shortcuts in newly generated graph.

Goal Directed Techniques. the principal idea lies in providing the search algorithm with the best possible estimate of how long it takes to reach the goal by choosing one or another way. Considering this estimate, the algorithm omits to visit locations that are unlikely to take part in the absolute shortest path and prefers these with higher potential. The biggest issue in this field is to generate the most accurate estimate in reasonable time. The complexity of these estimates or heuristics varies from the most trivial ones using euclidean distance through precomputing shortest paths between some specific spots (ALT, Precomputed cluster distance [38]) up to more elaborate ones trying to take benefit from the topology of the road network (Geometric containers [46, 56], Arc flags).

Contraction Hierarchies (CH). This technique is based on an assumption that sufficiently long paths converge to a small network made of highways or first-class roads. Therefore, it is beneficial to construct a graph enriched by shortcuts (joined locations) and run planning algorithms taking benefit of these auxiliary edges, which significantly shortens the final query time. CHs are a two-phase algorithm. During the first preprocessing phase, vertices or locations that are not connected by any road are contracted by creating shortcuts between them. The product of this procedure is the contracted road network (contracted graph), which is used during the second query phase to find the shortest path between two given locations. There are also several associated subproblems, such as choosing vertices (locations) to be contracted in the preprocessing phase which is studied for example in [22, 11, 2, 33]. The goal is to minimize the number of produced shortcuts, leading to a sparser contracted graph, shorter times of preprocessing phase, and more minor memory requirements.

As shown in experiments performed on region of Western Europe presented in [4], using the most advanced Arc flags technique the query time drops from Dijkstra's more than 2000000 μs to slightly more than 400 μs which represents an enormous improvement. In the same experiment CH even outperform Arc flags by query times around 100 μs.

1.3.2 Routing for Electric Vehicles

There has already been a significant effort spent on the problem of EV routing and, similarly, on the planning of charging stops. In the very beginning, considerable attention was paid to the planning of energy-efficient routes minimizing the consumption as a single criterion [14, 45, 7, 62]. As authors in [6] claim and our experience confirm that as well, such an approach often leads to prolongations in tens of minutes even on paths within the Czech Republic compared to paths retrieved by time optimizing routing algorithms.

This is caused by forcing the planning algorithm to use routes with lower speeds, which leads to lower consumption of electric energy. Such adverse effects are even more severe when considering longer driving distances.

Jochen Eisner et al. [14] use routing graph capturing the road network with energy consumption assigned to its edges to search for the sequence of vertices connected by edges such that in any vertex of the sequence, there is no occurrence of running out of energy, and it minimizes the total energy consumption. They propose using a modification of the Bellman-Ford algorithm, where the cost of individual edges is assigned during the runtime of the algorithm. This is done by a function modeled to avoid overcharging as a result of recuperation and also to make all edges, where running out of energy would occur, infeasible by setting their cost to infinity. On the other hand, they also propose to use a shifting idea introduced by [32] to be able to shift negative costs in the graph without negative cycles. As a consequence of this procedure, they can use Dijkstra's algorithm enriched by Contraction Hierarchies preprocessing technique to achieve final query times around 200 ms on the road network capturing parts of Germany.

Another approach worth mentioning is modeling the problem as the Constrained Shortest Path (CSP) problem [25] searching for the shortest path in either consumption or time criteria under the constraint of not running out of energy or not overcoming a specific time limit [48, 50]. On the contrary to [14], where the mechanism of charging was utterly absent, in [48] written by Sabine Storandt, charging stops are modeled by loading stations, where EV recharges to its full SoC whenever it visits the corresponding vertex in the planning graph. Storandt also introduces an additional distance function that assigns its estimated driving time to each edge in the graph. Consequently, they can model both variants, namely optimization of the consumption with time constraints or the opposite optimization of the driving time under the SoC limitations. Bicriteria Label setting variant of Dijkstra's algorithm for the SP is used to solve the problem. Each label consists of a tuple containing EVs SoC when arriving at a given vertex in the graph and the length of the current path (time). As for heuristic techniques, preprocessing of the shortest feasible paths between charging stations appears for the first time in this survey paper. Furthermore, authors construct the auxiliary graph consisting exclusively of loading stations and the shortest feasible paths between them in the query time. That is achieved by searching for vertices reachable from a given starting vertex together with vertices from which it is possible to reach the goal. These two sets intersect with the set of charging stations which already implies the resulting auxiliary graph used for routing.

Significant limitations of methods presented in these mentioned articles are severe simplifications in modeling charging stops or energy consumption which is often modeled by a linear function, and only charging to 100 % is considered. Eisner et al. [14] even omit the mechanism of charging entirely despite the fact that proper estimation and modeling of such features may have a crucial impact on the implementation of proposed algorithms in production-based systems.

To the best of our knowledge, the most advanced techniques, methods, and algorithms in planning for EVs were introduced in several consecutive articles written by the team of Dr. Dorothea Wagner [5, 9, 6, 8]. Throughout their work, they define the problem of routing electric vehicles and planning charging stops as a CSP problem searching for the shortest path, optimizing the sum of total travel and charging times constrained by the starting SoC, which can not be completely drained during the journey. They also model charging curves of electric vehicles by piecewise linear and concave functions, which are by far closer to the real-world situation than approximations by linear functions used in [48, 35]. The approach of Baum et al. [6] consists of several components. The core of their CHarge algorithm presented in the paper is formed by the Charging Function Propagation Algorithm (CFP), which is based on the ordinary Bicriteria Dijkstra's algorithm similar to the one in [48].

Charging Function Propagation Algorithm – CFP. the optimization algorithm searches for the shortest plan (time criteria) such that the EV consumption does not exceed the starting SoC ($\mathcal{NP}$-hard). It operates with labels containing indicators of the search state such as current time spent on the route, current location, last visited charging station, or SoC when arriving in the last charging station. These labels are organized in two sets called *settled* and *unsettled*. In each algorithm iteration, one label is extracted from the *unsettled* set of labels, which is implemented as a priority queue (ordered by the current time spent on the route). The label is then added to *settled* labels set, and new labels induced by the extracted one are produced. The first time label being allocated in the destination vertex is extracted from the queue of *unsettled* labels, the result is found, and final plan is constructed. A substantial fragment of CFP is an incorporation of charging actions into the planning process by producing a set of new labels, each capturing a specific charging time in the previous charging station each time a label allocated in the next charging station is extracted. This enables a reconstruction of the final path with a concrete assignment of charging stops and their duration, minimizing the overall time of the trip. Furthermore, due to the piecewise linearity and concavity of charging curves, Baum et al. proved that the minimal number of labels capturing charging actions at the previous charging station is limited by the number of breakpoints of charging curves while preserving the optimality of the algorithm.

Besides the main planning algorithm, several admissible and as well sub-optimal heuristics together with an implementation of CH [21] technique into the algorithm are introduced in [6]. These heuristics are based mainly on estimating driving and charging time and pruning the planning graph to save time in the query phase. Joining the work together, they achieved query times in seconds on the continental-scale regions while preserving the high level of natural world conditions by including all types of charging stations.

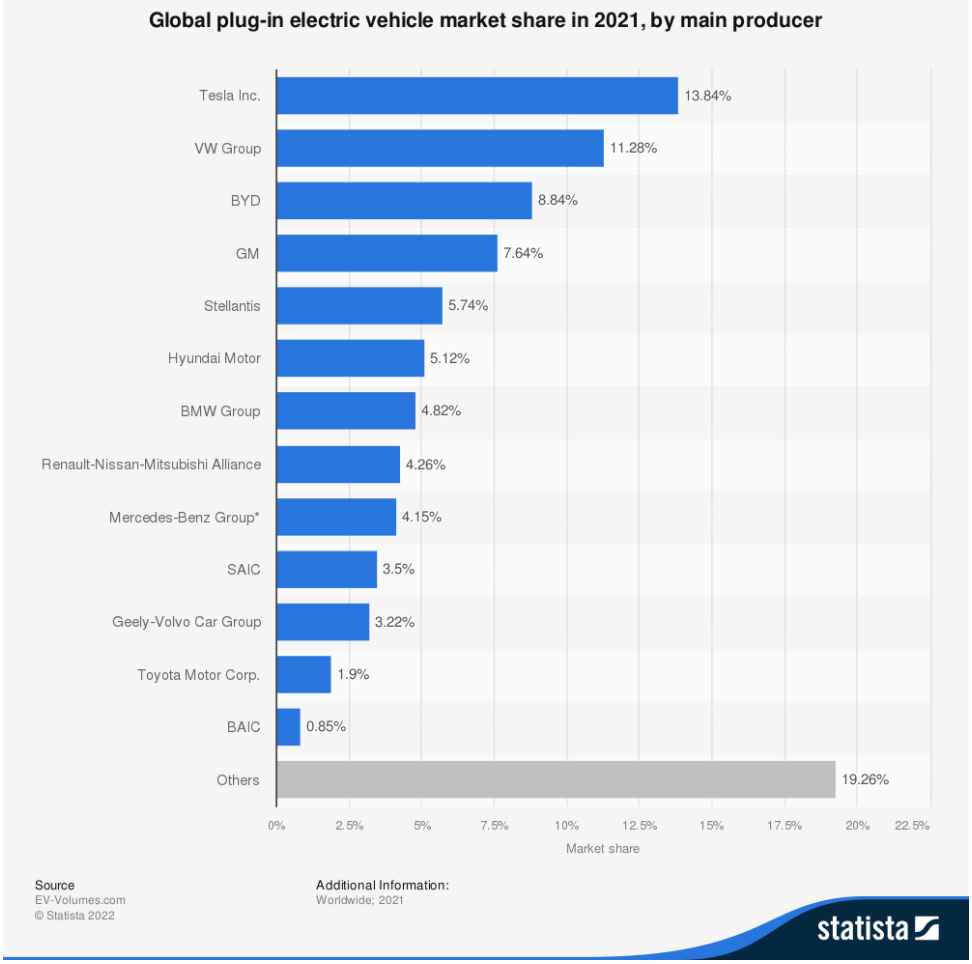


Figure 1.1: Global plug-in electric vehicle market share in 2021 [17].

1.3.3 Current Electric Vehicle Navigation Systems

Based on [61], the number of EVs in Europe has already overcome five million registered units by the end of 2021. An estimate was set at 17.5 million [59] considering the whole world. Such an amount of users automatically forces manufacturers of EVs to equip their products not only with wheels, steering wheels, batteries, and radio but also with a navigation system that helps its drivers plan their routes.

Tesla. According to the chart in Figure 1.1 published on the server Statista, Tesla was the leading company in sales of EVs in 2021, forming almost 14 % of the whole market. Another chart in Figure 1.2 shows that annual sales of Tesla vehicles rapidly increased every year from 2016 to 2021. Considering the usual claim that Tesla is primarily a software and artificial intelligence company with its enormous effort in the self-driving field, it is not a surprise that its navigation system is among the best drivers can get on the market. According to accessible online sources [51] documenting the functionality of

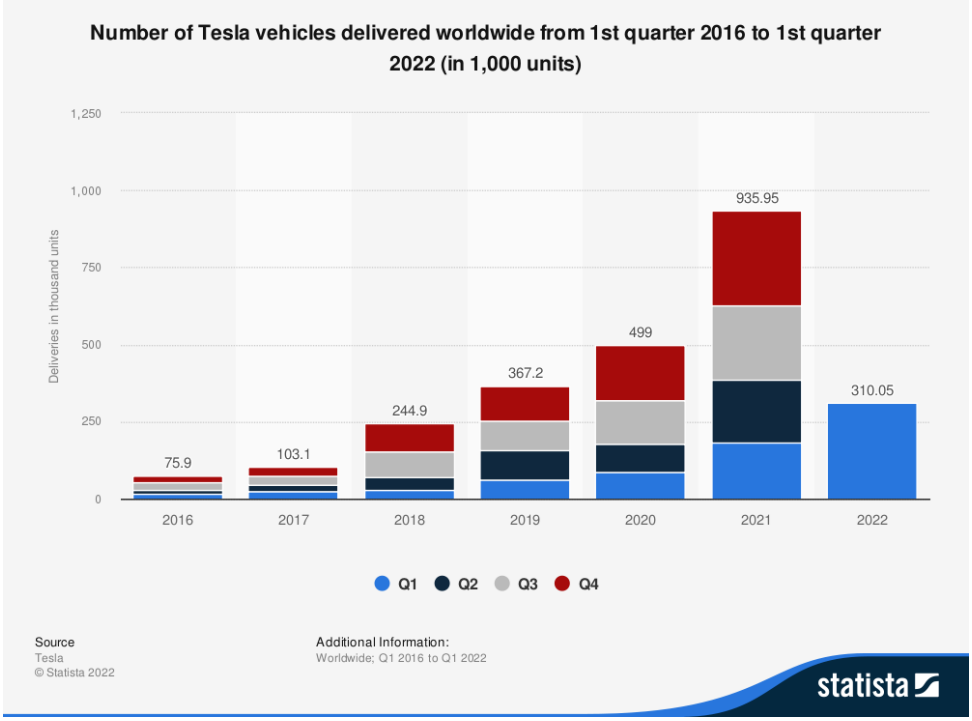


Figure 1.2: Number of Tesla vehicles delivered worldwide from 1st quarter 2016 to 1st quarter 2022 [52].

integrated navigation in Tesla vehicles, users are served with a variety of possible settings. From the ordinary route planning considering the current traffic information, displaying available charging locations of all distinct speeds, prediction of energy usage on the route taking into account elevation, driving style or weather up to the functionality they call *Trip planner*. The description of *Trip planner* fits the most to the specification of problems discussed in this thesis. Tesla claims that using this feature, users can plan longer road trips as the *Trip planner* provides them the plan with charging stations as intermediate destinations together with the recommended time to stay charging there. The most significant limitation of this solution is that plans include only superchargers. Therefore, produced plans may suffer from non-optimality in the case of areas with a sparse density of this type of chargers. Another limitation of generated plans is the fact that according to [43] the number of charging stops is minimized instead of the total driving time, which can lead to prolongations of routes since different criteria is optimized. On the other hand, Tesla reports the possibility of showing all nearby charging stations (including slow ones). This way, they leave the responsibility of planning charging events using these slower stations to drivers. Another feature Tesla supports is the possibility to navigate to favorite restaurants or other POIs by their *I'm Feeling Lucky, Hungry* feature. However, the limitation seems to be that they can only suggest these locations independently and then perform the navigation. On the contrary to this, we focus on the direct implementation of POIs into the optimization process.

Mercedes. An application called *Mercedes me Charge* [39] is provided by Mercedes to its customers. It supports the possibility of searching for the closest charging station together with a piece of information capturing their current availability. Using the same application, drivers can plan long-range trips with integrated charging stations and several possible adaptations, such as minimal arriving SoC at charging stations or at the destination. The official Mercedes web page states that the routing algorithm prioritizes rapid charging stations and considers current SoC, weather, charging infrastructure, and route topology.

Hyundai Kia Automotive Group. Hyundai provides their customers *Charge myHyundai* application [29] which enables them to search for chargers together with their availability in the region of the whole Europe. Nevertheless, it does not support the planning of trips at all. It does not even seem to prove smooth running since the algorithm clustering available chargers suffered from a fatal lag during the whole testing process of the application. According to found video reviews [47, 42], infotainment installed inside Hyundai EVs does not provide any route planning functionality with integrated charging stations as well.

Volkswagen Group. As shown in Figure 1.1, Volkswagen Group covering brands such as Volkswagen, Audi, Porsche or Škoda Auto, took second place in the overall market with plug-in EVs in 2021. Such a position suggests the best chance to develop a navigation system supporting users of EVs produced by these companies on their journeys. Unfortunately, online sources and applications publicly available on official websites of these automotive giants do not show they fulfill this great potential. Volkswagen as a brand supplies their EVs by a system called *Electric Vehicle Route Planner*. A limited version of this software is even available online [54]. Unfortunately, the number of parameters to be set is limited merely to the start and destination location, the possibility of choosing the concrete EV model and two charging station selection strategies. Navigation integrated directly in EVs supports more possible adaptations such as setting of the minimal destination SoC. Nevertheless, many user videos and discussions declare its poor functionality manifesting itself in an inability to produce plans within some European cities. Very rough and impractical parameter setting, absence of possibilities to customize plans, or even buggy behavior are reported as well [55, 30].

Škoda Auto provides an application capable of searching for the shortest path listing all available charging stations along with it on their web [63]. Even though an official infotainment presentation video of Škoda Enyaq iV [44] shows the possibility of planning long routes together with integrated charging stations, user experiences to be found online prove its poor performance in producing plans far from optimum or not returning plans at all [16].

A Better Route Planner. Throughout the whole research process of the current EV navigation systems supplied by automotive companies, we were constantly coming across the platform called *A Better Route Planner* [31].

An abbreviation ABRP resonates in most discussions led by EV enthusiasts or even reviews of systems provided by automotive manufacturers. These facts support the claim that the EV community considers ABRP to be the most reliable, flexible, personable, accurate, and user-friendly alternative [43]. It provides an exhaustive number of settings, including many supported vehicles, initial and destination SoC, real-time traffic, weather, charger types, preferred length of charging stops, or even the possibility to schedule stops for food or shopping along the generated path. Besides minor UI issues and minor feature requests, ABRP seems to be the closest to an ideal navigation system for planning routes for EVs and the most severe commercial competitor for comparison with the results achieved in this thesis.

1.4 Scope of the Thesis and Contribution

This thesis focuses on research, formalization, development, implementation, and evaluation of route planning algorithms with integrated planning of charging stops and the possibility to customize and enrich generated plans concerning the driver's requirements. Namely, it focuses on integrating POIs into the planning algorithms, scalability of result solutions, personification performed by constraints integration and consideration of real-world conditions.

The main contributions of the thesis are:

1. A formal definition of the problem of Route Planning for Electric Vehicles with Charging Stops and Points of Interest.
2. Design of methods and techniques for constructing an auxiliary graph from the route graph. Thanks to this, the result planning algorithm operates on a significantly sparser graph, which leads to shorter query times. The principal of these techniques rests in:
    a. Using SP problem solutions between charging stations as edges in the auxiliary graph.
    b. Introducing the heuristic approach of charging stations selection individual for each planning query taking into account its localization.
3. Introduction of an Integer Linear Programming (ILP) model solving the defined problem serving as a baseline solution.
4. Design and adaptation of algorithms based on the current state of the art solution presented in [6] by introducing generalizations in enabling the incorporation of additional constraints in the form of scheduling user-specified actions of fixed length (e.g. amenity stops nearby charging stations) and still optimizing the total time of the plan.
5. Elaboration of EV consumption and charging model, which crystallize from the current literature and correspond to the real-world conditions.

6. Evaluation of proposed solutions by direct results comparisons with the ILP baseline model and a discussion of differences, advantages, and disadvantages of the proposed approach in confrontation with the current state of the art and solutions available in automotive industry.

Chapter 2

Problem Statement

In this chapter, we formally define the problem of Route Planning for Electric Vehicles with Charging Stops and Points of Interest (EVRT-POI) in its full generality. EVRT-POI searches for a path between two locations while minimizing the total time spent driving, charging, and scheduling POIs. Additionally, such a path has to take battery constraints into account, i.e., the EV can not run out of energy on the path, and all POIs have to be scheduled respecting the desired frequency and duration of its stops.

2.1 List of Used Symbols and Notation

We introduce a notation used later in the thesis in a formal definition presented in Section 2.2 and solution approach delivered in Chapter 3. It is divided into three logical sections to support the clarity.

Route Planning for Electric Vehicles

- $G = (V, E)$ – a directed graph capturing the road network.
- V – a finite set of vertices.
- $E \subseteq V \times V$ – a finite set of edges (u, v), where $u, v \in V$.
- $P_{s,t} \in V^m$ – a path $P_{s,t} = [s = v_1, \ldots, t = v_m]$ consisting of $m \in \mathbb{N}$ vertices starting in $s \in V$, ending in $t \in V$, passing through $v_2, \ldots, v_{m-1}$, where $v_i \in V$, $\forall i \in \{2, \ldots, m-1\}$ and $(v_i, v_{i+1}) \in E$, $\forall i \in \{1, \ldots, m-1\}$.
- $\mathbf{d} : E \to \mathbb{R}_{\geq 0}$ – a time cost function associated with each edge in the graph assigning the time needed to traverse it measured in minutes. In the following text we use a simplified notation $\mathbf{d}(u, v)$ instead of $\mathbf{d}((u, v))$ to express a time to traverse the edge $(u, v) \in E$. The same applies to the following functions.
- $\mathbf{c} : E \to \mathbb{R}$ – a consumption cost function associated with each edge in the graph assigning an energy of EV battery capacity consumed by the edge traversal measured in percent.

- $\mathbf{l} : E \to \mathbb{R}_{\geq 0}$ – a distance function associated with each edge in the graph assigning a distance measured in meters.
- $\mathbf{D} : V^m \to \mathbb{R}_{\geq 0}$ – a driving time function assigning the total driving time in minutes to the path $P_{s,t} \in V^m$ with its prescription:

$$\mathbf{D}(P_{s,t}) = \sum_{i=1}^{m-1} \mathbf{d}(v_i, v_{i+1}).$$

- $\mathbf{C} : V^m \to \mathbb{R}$ – a consumption function assigning the total consumption measured in percent of EV battery capacity to the path $P_{s,t} \in V^m$ with its prescription:

$$\mathbf{C}(P_{s,t}) = \sum_{i=1}^{m-1} \mathbf{c}(v_i, v_{i+1}).$$

- $\mathbf{L} : V^m \to \mathbb{R}$ – a distance function assigning the total distance in meters to the path $P_{s,t} \in V^m$ with its prescription:

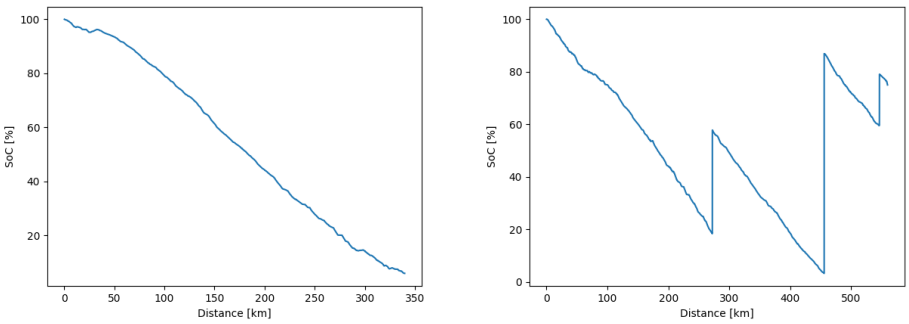
$$\mathbf{L}(P_{s,t}) = \sum_{i=1}^{m-1} \mathbf{l}(v_i, v_{i+1}).$$

- $\beta_s \in [0, 100]$ – an initial SoC measured in percent of the EV battery capacity at the starting vertex $s \in V$.
- $\beta_t \in [0, 100]$ – a minimal destination SoC measured in percent of the EV battery capacity at the target vertex $t \in V$.
- $\beta_{\text{safe}} \in [0, 100]$ – a minimal SoC measured in percent of the EV battery capacity at any vertex $v \in V$. Serves for modeling of safety margins.
- $\boldsymbol{\beta}_{\text{min}} : V \to [0, 100]$ – a function mapping all vertices $v \in V$ to the value of the minimal SoC in v. It is defined as follows:

$$\boldsymbol{\beta}_{\text{min}}(v) := \begin{cases} \beta_s & \text{if } v = s \\ \beta_t & \text{if } v = t \\ \beta_{\text{safe}} & \text{otherwise.} \end{cases}$$

- $M \in \mathbb{R}_{\geq 0}$ – a maximal capacity of the EV battery measured in kWh.
- $\mathbf{p} : V^m \times [0, 100] \to \mathbb{R}^m$ – a function mapping path $P_{s,t} \in V^m$ and initial SoC $\beta_s \in [0, 100]$ to a consumption profile $\mathcal{P} = [\beta_1, \ldots, \beta_m]$, where $\beta_i \in \mathbb{R}$, $i \in \{1, \ldots, m\}$ are SoC values corresponding to individual vertices in the path $P_{s,t}$. $\beta_1 = \beta_s$ and β_i for $i \neq 1$ are calculated as a differences between the initial β_s and the energy consumption of sub-paths from s to v_i, which can be written as:

$$\beta_i = \beta_s - \mathbf{C}(P_{s,v_i}) = \beta_s - \sum_{i=1}^{i-1} \mathbf{c}(v_i, v_{i+1}).$$



(a) : Consumption profile of a path $P_{s,t}$.

(b) : Augmented consumption profile of a path $P_{s,t}$.

Figure 2.1: Consumption profile examples.

An example of such a consumption profile is marked in Figure 2.1a, where on the x axis, there is the distance of individual vertices from the start in km, and on the y axis, there is a corresponding SoC measured in percent for each vertex.

- $\mathbf{hv} : V \times V \to \mathbb{R}_{\geq 0}$ – a function mapping two vertices $u, v \in V$ to the haversine distance between them measured in meters. The haversine distance determines the distance of two locations on a sphere, given their coordinates. Its computation is described in [60].
- $\mathbf{hv}_{\min} : V^m \times V \to \mathbb{R}_{\geq 0}$ – a function mapping a path $P_{s,t} \in V^m$ and a vertex $u \in V$ to the haversine distance between u and vertex $v \in P_{s,t}$ with the shortest haversine distance $\mathbf{hv}(u, v)$:

$$\mathbf{hv}_{\min}(P_{s,t}, u) = \min_{v \in P_{s,t}} \mathbf{hv}(u, v).$$

the distance $\mathbf{hv}_{\min}$ is considered as the distance between the vertex u and the path $P_{s,t}$.

Planning of Charging Stops

- $S \subseteq V$ – a finite set of charging stations as a subset of vertices $v \in V$.
- $\mathbf{cf}_v : \mathbb{R}_{\geq 0} \to [0, 100]$ – a charging function for each charging station $v \in S$ mapping charging time $t \in \mathbb{R}_{\geq 0}$ to the resulting SoC $\beta_{out} \in [0, 100]$ when charging from zero percent. For the simplicity of the following definitions, we also define charging functions for $v \notin S$ as an identity function which means that they return 0 regardless of the given value of t. Examples of several charging functions differing in their speed are shown in Figure 2.2a. On the x axis, there is the time in minutes spent by charging, and on the y axis, there is an amount of energy charged measured in percent in case of charging from 0 %.

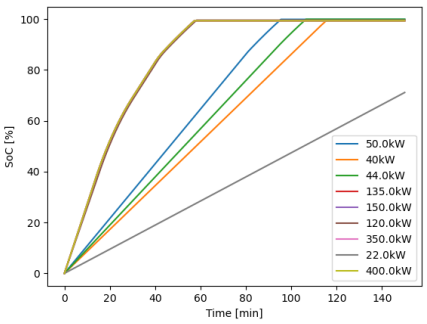
- $\mathbf{cf}_v^{-1} : [0, 100] \to \mathbb{R}_{\geq 0}$ – an inverse charging function for each charging station $v \in S$ mapping a target SoC $\beta_{out} \in [0, 100]$ to a charging time $t \in \mathbb{R}_{\geq 0}$ which is needed to spent by charging in v to reach SoC β_{out} in the case of charging from zero percent. Examples of several inverse charging functions differing in speed of charging stations are shown in Figure 2.2b. On the x axis, there is an amount of energy to be charged measured in percent, and on the y axis, there is a time needed to spend by charging measured in minutes in case of charging from zero percent.

- $T_{P_{s,t}} \in \mathbb{R}^m_{\geq 0}$ – a charging time profile of a path $P_{s,t} \in V^m$ in a form of $T_{P_{s,t}} = [t_1, \ldots, t_m]$, where $t_i = 0$ for $v_i \in P_{s,t}$ and at the same time $v_i \notin S$ and $t_i \in \mathbb{R}_{\geq 0}$ for $v_i \in S$. Written in words, t_i is a time spent by charging in each vertex $v_i \in P_{s,t}$ which is equal to 0 in the case that v_i is not a charging station and it can acquire non-zero values in the case a corresponding v_i is a charging station.

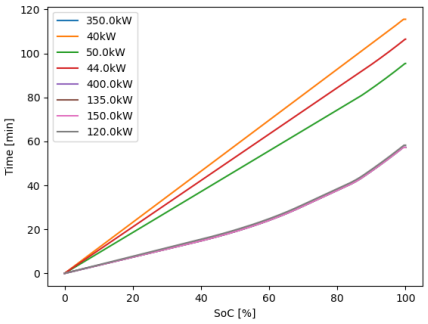
- $\mathbf{h} : V^m \times \mathbb{R}^m_{\geq 0} \times [0, 100] \to \mathbb{R}^m$ – a function mapping $P_{s,t} \in V^m$, charging time profile of the path $T_{P_{s,t}} \in \mathbb{R}^m_{\geq 0}$ and an initial SoC $\beta_s \in [0, 100]$ to a consumption profile $\mathcal{H} \in \mathbb{R}^m$, $\mathcal{H} = [h_1, \ldots, h_m]$ which corresponds to a consumption profile $\mathbf{p}(P_{s,t}, \beta_s)$ augmented by an amount of energy charged during the path in corresponding vertices $v \in P_{s,t}$. Specific values $h_i \in \mathbb{R}$, $i \in \{1, \ldots, m\}$ are then retrieved by induction as:

$$h_i := \begin{cases} \min\{\mathbf{cf}_s(t_i + \mathbf{cf}_s^{-1}(\beta_s)), 100\} & \text{if } i = 1 \\ \min\{\mathbf{cf}_{v_i}(t_i + \mathbf{cf}_{v_i}^{-1}(h_{i-1} - \mathbf{c}(v_{i-1}, v_i))), 100\} & \text{if } i > 1. \end{cases}$$

  Augmented SoC value h_1 for the starting vertex $s \in V$ is either a final SoC after charging time t_1 from initial SoC β_s or 100 % in the case that t_1 is larger than time needed to charge to 100 % at a charging station located in s if there is one. As functions $\mathbf{cf}_v$ and $\mathbf{cf}_v^{-1}$ are defined to capture charging from 0 %, to receive the target value of SoC after charging time t_i from initial SoC β_s we firstly compute the time needed to charge from 0 % to β_s by $\mathbf{cf}_s^{-1}(\beta_s)$, add this time to desired charging time t_i and plug the result back into the $\mathbf{cf}_v$ function which provides the correct resulting SoC. Each other value t_i, $i \in \{2, \ldots, m\}$ is equal to 100 % or a final SoC after charging time t_i from the initial SoC equal to the value of SoC when leaving the previous v_i decreased by a consumption $\mathbf{c}(v_{i-1}, v_i)$ of the edge (v_{i-1}, v_i) which is the value of SoC when entering v_i. The principal of computing this value is the same as in the case of h_1. An example of such an augmented consumption profile is in Figure 2.1b, where steps around 290 km, 460 km and 560 km on the x axis sign three charging actions.



(a) : Charging functions cf_v for chargers with distinct charging speeds.



(b) : Inverse charging functions cf_v^{-1} for chargers with distinct charging speeds.

Figure 2.2: Charging function examples.

Planning of POIs

- $\mathbf{hv}_{\text{ch}} : V \to \mathbb{R}_{\geq 0}$ – a function mapping a vertex $u \in V$ to the haversine distance measured in km from the closest charging station $v \in S$:

$$\mathbf{hv}_{\text{ch}}(u) = \min_{v \in S} \mathbf{hv}(u, v).$$

- $\epsilon \in \mathbb{R}_{\geq 0}$ – a maximal distance of a concrete POI from the closest charging station $v \in S$.

- $L \subseteq V$ – a subset of vertices $v \in V$ expressing the possibility of scheduling POI stop in vertices $v \in L$. Considering, for example, shopping as a kind of activity to be done in POIs, the subset L captures available shops located in vertices $v \in V$. However, except for accepting all vertices associated with a POI of the particular type, only vertices satisfying $\mathbf{hv}_{\text{ch}}(v) \leq \epsilon$ are included to ensure the possibility of parallel charging and performing scheduled activity in the POI vertex $v \in L$.

- $d_{\text{POI}} \in \mathbb{N}$ – a duration period measured in minutes determining how long should each stop for POI take when scheduled.

- $f_{\text{POI}} \in \mathbb{N}$ – a frequency determining how often should be the stop for POI scheduled. The meaning of values f_{POI} and d_{POI} is that a POI has to be scheduled at least once per f_{POI} minutes for a duration of d_{POI} minutes.

- $Z_{P_{s,t}} \in \{0,1\}^m$ – assuming path $P_{s,t} \in V^m$ of the length m, $Z_{P_{s,t}}$ represents a schedule of POIs in vertices $v \in P_{s,t}$. $Z_{P_{s,t}} = [z_1, \ldots, z_m]$, where $z_i = 0, \forall i : v_i \notin L$ and $z_i \in \{0,1\}, \forall i : v_i \in L$ which signifies that it is not possible to schedule POI in place which is not suitable for this purpose. If $z_i = 0$ it means that POI is not scheduled in the corresponding vertex $v_i \in P_{s,t}$, otherwise it is scheduled there and it takes d_{POI} minutes.

- $\mathbf{w} : \{0,1\}^m \times \mathbb{N} \to \mathbb{N}$ – a function mapping a POI schedule $Z_{P_{s,t}}$ and an index $i \in \{1,\ldots,m\}$ to an index of the closest previous occurrence of value 1 in $Z_{P_{s,t}}$ for given index i i.e., an index of the vertex $v \in P_{s,t}$, where POI was scheduled last time. The function is formally defined as follows:

$$\mathbf{w}(Z_{P_{s,t}}, i) := \begin{cases} 1 & \text{if } \sum_{j=1}^{i-1} = 0 \\ \sum_{l=i-1}^{1} \operatorname{sgn}(\sum_{m=l}^{i-1} z_l) & \text{if } \sum_{j=1}^{i-1} \neq 0 \end{cases}.$$

  It returns 1 if the POI is not scheduled in any vertex preceding the ith vertex. Otherwise, it calculates an amount of sub-intervals ending by the index equal to $i-1$ containing at least one occurrence of POI scheduled, which is, at the same time, equal to the index of the previous occurrence of the POI in $Z_{P_{s,t}}$.

- $\mathbf{x} : V^m \times \mathbb{R}^m_{\geq 0} \times \{0,1\}^m \to \mathbb{R}^m_{\geq 0}$ – a function mapping a path $P_{s,t} \in V^m$ of a length m, its charging profile $T_{P_{s,t}} \in \mathbb{R}^m_{\geq 0}$ and a schedule $Z_{P_{s,t}} \in \{0,1\}^m$ of POIs in vertices $v \in P_{s,t}$ to a vector $X \in \mathbb{R}^m_{\geq 0}$, $X = [x_1,\ldots,x_m]$. The resulting vector X contains times passed from the last occurrence of POI scheduled on the path $P_{s,t}$ in each particular vertex $v \in P_{s,t}$. $x_1 = 0$ and the formula for the rest of values $x_i \in X$, $i \in \{2,\ldots,m\}$ is:

$$x_i = \mathbf{D}(P_{v_{\mathbf{w}(Z_{P_{s,t}},i)}, v_i}) + \sum_{j=\mathbf{w}(Z_{P_{s,t}},i)+1}^{i-1} \max\{t_j, z_j \cdot d_{\text{POI}}\}.$$

  Each x_i value consists from the driving time on the sub-path $P_{v_{\mathbf{w}(Z_{P_{s,t}},i)}, v_i}$ from the last occurrence of POI on the path $P_{s,t}$ to the current vertex v_i and the sum of charging times or times spend by stopping in POI on the sub-path $P_{v_{\mathbf{w}(Z_{P_{s,t}},i)+1}, v_{i-1}}$ leading from the vertex following the vertex $v_{\mathbf{w}(Z_{P_{s,t}},i)}$ to the vertex preceding the v_i.

2.2 Problem Statement

Having defined all previous, we can formalize the following EVRT-POI optimization problem as follows:

Input Query

The query Q consists of the road graph graph G, starting and destination vertices s and t with corresponding SoC values β_s and β_t, the set of charging stations S, the set of POIs L, the POI stop duration d_{POI} and the frequency f_{POI}.

$$Q = (G, s, t, \beta_s, \beta_t, S, A, L, d_{\text{POI}}, f_{\text{POI}}).$$

Output Plan

The final plan R consists of the path $P_{s,t}$, the charging profile $T_{P_{s,t}}$ and the schedule for POIs $Z_{P_{s,t}}$.

$$R = (P_{s,t}, T_{P_{s,t}}, Z_{P_{s,t}}).$$

Problem Formulation

$$\underset{P_{s,t}, T_{P_{s,t}}, Z_{P_{s,t}}}{\operatorname{argmin}} \quad \mathbf{D}(P_{s,t}) + \sum_{i=1}^{m} \max\{t_i, z_i \cdot d_{\mathrm{POI}}\}$$

subject to:

$$\begin{array}{lllll}
(\mathrm{I1}): & & \beta_1 = \beta_s & \beta_1 \in \mathbf{p}(P_{s,t}, \beta_s) \\
(\mathrm{I2}): & \forall i \in [2, m] & h_i \geq \boldsymbol{\beta}_{\min}(v_i) & h_i \in \mathbf{h}(P_{s,t}, T_{P_{s,t}}, \beta_s) \\
(\mathrm{I3}): & \forall i \in [1, m] & x_i \leq f_{\mathrm{POI}} & x_i \in \mathbf{x}(P_{s,t}, T_{P_{s,t}}, Z_{P_{s,t}})
\end{array}$$

Starting the description from the criteria function, the problem searches for the path $P_{s,t}$, the charging profile $T_{P_{s,t}}$ and the POI schedule $Z_{P_{s,t}}$ which are minimizing the total time of the plan. It consists of the total driving time $\mathbf{D}(P_{s,t})$ and the time spend by charging or time in POIs which is captured by the sum in the second part of the function. The problem is constrained firstly by (I1)–(I2) which gradually express that the initial SoC $\beta_1 \in \mathbf{p}(P_{s,t}, \beta_s)$ has to be equal to the given value β_s, and there is no occurrence of running out of energy in any vertex of the path $P_{s,t}$. By not running out of energy we mean that all values $h_i \in \mathbf{h}(P_{s,t}, T_{P_{s,t}}, \beta_s)$ are above $\boldsymbol{\beta}_{\min}(v_i)$, which is equal to β_{safe} for vertices v_i, $i \in \{2, \ldots, m-1\}$ and it is equal to β_t for vertex $v_m = t$.

The second set of constraints consisting of the single constraint (I3) ensures that there is no such a moment during the execution of the final plan R in which the interval between two consecutive occurrences of POI stops would exceed the desired value f_{POI}.

2.3 Complexity of the Problem

Similarly to the problem introduced in [6], it is possible to reduce the Constrained Shortest Path (CSP) problem [25] with two non-negative criteria functions searching for the shortest path in the first criteria constrained by a specific value of the second criteria to our EVRT-POI problem. We can perform the transformation by considering empty sets of charging stations ($S = \emptyset$), no POIs to be scheduled, and non-negative consumption function $\mathbf{c}$. Due to this, we can transform the described CSP problem into our EVRT-POI and solve it by the methods proposed in the thesis. As a decision variant of this problem is listed in $\mathcal{NP}$-complete problems in [27, 58], our problem is $\mathcal{NP}$-hard.

Chapter 3

Solution Approach

Due to our primary focus on scalability and satisfying the real-world requirements of EV drivers, we were forced to rethink the importance of time criteria optimization. All survey papers presented in Chapter 1.3 considering similar problem definitions to ours (e.g., [9, 6, 48]) put the overall time of the plan with integrated charging stations on the first place in a priority. We focus more on customizability, flexibility, and time criteria minimization. This enables direct modeling of driver's requirements such as the generation of the fastest possible plan, energy-saving plan, cost-saving plan, and charging stops minimizing plan.

The intention which accompanies us during the whole process of solution proposal is to verify whether it is possible to introduce a set of algorithms solving the EVRT-POI problem, which heavily uses results of the simple shortest path problem while preserving reasonable relaxations. The motivation for this is to use benefits from years of research in the field of shortest paths in solving fairly more complex EVRT-POI problem. Using this premise, we define the final solution consisting of two isolated parts: auxiliary graph construction and optimization algorithm. Both can be further divided into several standalone subproblems that support the overall versatility of the solution and provide a space for trade-offs between final query times and an amount of relaxed and heuristic conditions on distinct parts of the problem.

Approaches presented in Section 1.3 often use the road graph in the full extent (or enriched by CH shortcuts [6, 8]). We introduce the concept of an auxiliary graph consisting of precomputed paths between a heuristically chosen set of charging stations interconnected with the start and the destination location. To enable precise modeling of real-world characteristics of different EV models, we introduce the consumption model elaborated from the current literature together with the framework for simple charging function modeling by piecewise-linear functions similar to the one presented in Baum et al. [6]. These two components serve as an input for two proposed optimization algorithms solving the problem defined in Chapter 2. The first solution consists of an ILP model which does not consider scheduling of POIs. It serves as a baseline for the final Charging Function Propagating Algorithm with Points of Interest (CFP-POI) based on the CFP algorithm introduced in [6] with several simplifications and adaptations modeling extra

constraints for scheduling POIs. Besides the potential-based heuristic adopted from the original paper, we introduce a new approach inspired by ALT and Precomputed cluster distance techniques. Instead of precomputing simple shortest paths, we precompute times of whole plans within specific locations with charging time integrated and use these precomputed times as heuristic values for final queries.

3.1 Solution Outline

The proposed solution consists of the preprocessing and the query phase for solving the CFP-POI problem. All techniques in the outline are introduced later in Chapter 3.

Preprocessing Phase

1. The construction and cashing of the route graph G from datasets introduced in Section 2.1.
2. The precomputation of distance matrices between charging stations described in Subsection 3.2.3.
3. The precomputation of charging functions $\mathbf{cf}_v$ for all charging stations $v \in S$ introduced in Section 3.4.
4. The precomputation of plans for Precomputation-Based heuristic introduced in Subsection 3.5.2.

Query Phase

1. The received query Q.
2. The estimation of corridors introduced in Subsection 3.2.1.
3. The selection of charging stations to be included in the auxiliary graph introduced in Subsection 3.2.2.
4. The construction of the auxiliary graph introduced in Subsection 3.2.4.
5. The usage of one of optimization algorithms introduced in Subsections 3.5.1 and 3.5.2 and the production of the plan.

3.2 Auxiliary Graph Construction

The primary purpose of introducing the auxiliary graph is to reduce the size of the graph used by optimization algorithms and save computation time. Instead of searching for reachable sets of nodes as presented in [49], we construct the graph merely from initial starting and destination vertices and heuristically determined set of charging stations (Subsection 3.2.2). Charging

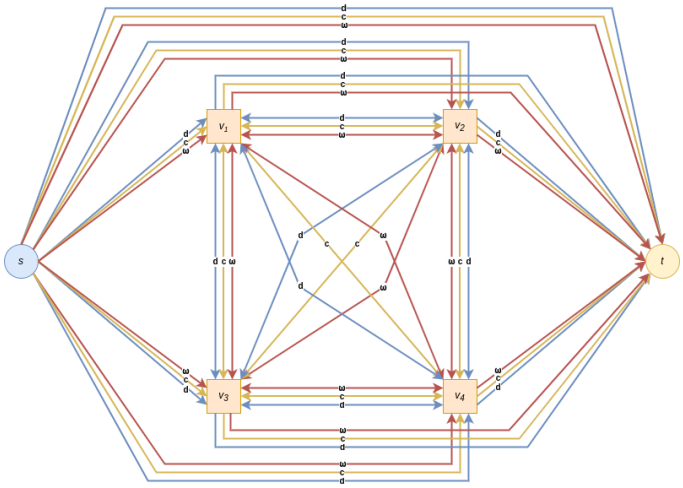


Figure 3.1: Schematic example of a full auxiliary graph (without final pruning).

stations are chosen concerning their speed and location relative to estimated final path corridors (Subsection 3.2.1). The auxiliary graph is a multi-graph consisting of a pruned complete graph between charging stations constructed from multi-criteria precomputed distance matrices (Subsection 3.2.3), multi-criteria edges from initial starting vertex s to all charging stations, and multi-criteria edges from all charging stations to the destination vertex t. A schematic example of an auxiliary graph is in Figure 3.1, where the blue circle stands for the start vertex s, the yellow circle stands for the destination vertex t, and orange squares are charging stations v_1, v_2, v_3, v_4.

Query $Q = (G, s, t, \beta_s, \beta_t, S, A, L, d_{\text{POI}}, f_{\text{POI}})$ is the input of the auxiliary graph construction procedure and the auxiliary graph $G_{\text{aux}} = (V_{\text{aux}}, E_{\text{aux}})$ with the set of charging stations S_{aux} are outputs.

3.2.1 Estimation of Corridors by the Shortest Paths with Multiple Criteria Functions

We estimate regions which the final $P_{s,t}$ path passes through. These regions are called corridors in the following text. Given an instance of the EVRT-POI problem and a general criteria function $\mathbf{q} : E \to \mathbb{R}$ assigning a criteria value to each edge $e \in E$, let the path $P^{\mathbf{q}}_{s,t} \in V^m$ be the shortest path from vertex s to t optimizing the function $\mathbf{q}$. The corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}} \subseteq V$ is then defined as a subset of vertices $u \in V$ whose haversine distance to at least one vertex $v \in P^{\mathbf{q}}_{s,t}$ is lower than ϵ:

$$\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}} = \{u \in V \mid \exists v \in P^{\mathbf{q}}_{s,t} : \mathbf{hv}(u, v) \leq \epsilon\}.$$

Such a corridor forms a belt of vertices surrounding the path $P^{\mathbf{q}}_{s,t}$ which is depicted in Figure 3.2 with $\epsilon = 50$ km (an amount of vertices $u \in \mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ is filtered in Figure 3.2 for visualization purposes). The main purpose of corridors $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ is to determine the area through which the final path $P_{s,t}$ passes and the area, where charging stations needed to drive long distances

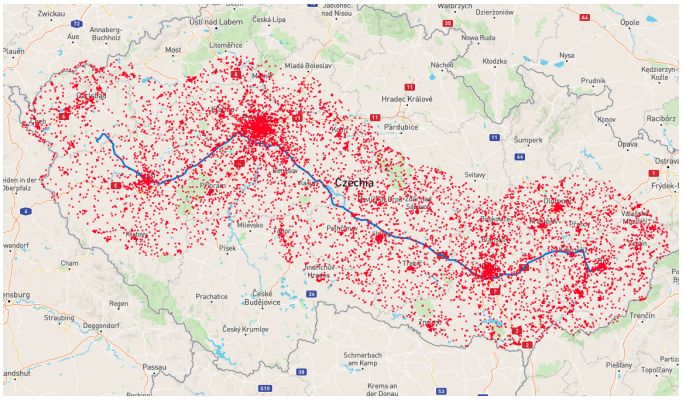


Figure 3.2: Example of corridor with $\epsilon = 50$ km.

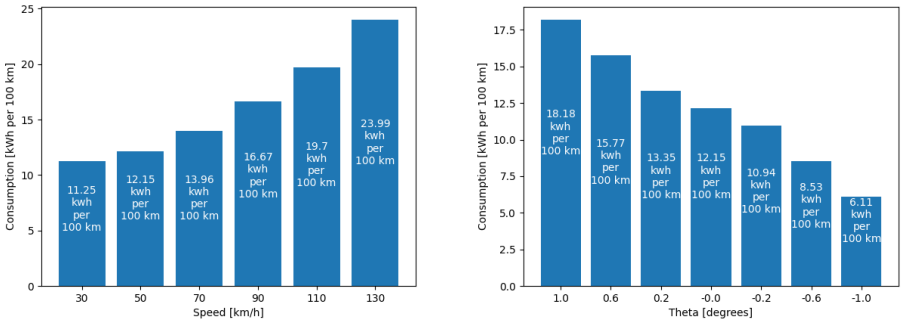
are located. Therefore, corridors $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ serve as an input into methods for choosing the final set of charging stations which form the auxiliary graph.

Criteria Functions

The shape of the corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ is determined by the shortest path $P^{\mathbf{q}}_{s,t}$ which is formed by the criteria function $\mathbf{q}$. Therefore, $\mathbf{q}$ is the key parameter in the estimation of the region for the final path $P_{s,t}$. The first proposed criteria function $\mathbf{q}$ is the driving time function $\mathbf{d} : E \to \mathbb{R}_{\geq 0}$ as defined in Section 2.1. Corridors $\mathcal{C}^{\epsilon}_{\mathbf{d}_{s,t}}$ then intuitively capture regions surrounding highways as the fastest possible roads. However, due to the set of consumption constraints (I1)–(I2) in the formulation of the EVRT-POI problem, it may happen that the shortest path optimizing the driving time does not correspond to the constrained shortest path optimizing the overall driving and charging time. For example, assuming the shortest path $P^{\mathbf{d}}_{s,t}$, it might happen that it is not possible to pass it without spending some additional time by charging as the energy consumption along the path is too high. On the other hand, there might be a different path $P^{\mathbf{q}}_{s,t}$ that takes a longer time, but as the amount of energy consumed on it is lower than in the case of $P^{\mathbf{d}}_{s,t}$, it does not require additional charging time, and it results in a shorter overall plan. Such a phenomenon is mainly caused by these three factors:

- Initial SoC β_s affects the optimal constrained shortest path. In the context of the previous example, it might happen that in the case of sufficient initial β_s even the path $P^{\mathbf{d}}_{s,t}$ is traversable without a charging stop and thought it might correspond with the optimal constrained shortest path.

- Consumption is heavily affected by speed. This can be seen in Figure 3.3a. It shows the dependency of the speed on the consumption considering a consumption model of big SUV Skoda Enyaq iV introduced in Section 3.3 with no elevation differences on the path and mild temperature values (5–15 °C). The consumption rises from 11.25 kWh per 100 km in the case of 30 km/h up to almost 24 kWh per 100 km,



(a) : Dependency of EV speed on consumption.

(b) : Dependency of elevation profile on consumption.

Figure 3.3: Factors affecting EV consumption.

considering speeds on highways. This may lead to invalidation of $P^{\mathbf{d}}_{s,t}$ in the case of insufficient β_s.

- Consumption is heavily affected by the elevation profile of the path. The proof of that is shown in Figure 3.3b. The consumption in kWh per 100 km constantly decreases with the decreasing slope of the road from a value of 17.5 kWh per 100 km up to 6.11 kWh per 100 km when the driving speed is constantly set at 70 km/h. We are again considering mild temperature values (5–15 °C) and technical parameters of Skoda Enyaq iV. Elevation may also cause a recuperation, state when an EV regenerates energy instead of consuming it when driving down the hill. An example of recuperation can be seen closely before 50 km in the consumption profile in Figure 2.1a as the consumption profile rises for a short amount of time. The effect of elevation may, for example, result in a situation when the shortest path $P^{\mathbf{d}}_{s,t}$ traverses through a highly situated mountain pass which disables the path in case of insufficient β_s. On the other hand, longer $P^{\mathbf{q}}_{s,t}$ leading around the mountain may still be feasible and hence result in the shorter plan.

As it is clear from previous paragraphs, taking into account merely driving time function $\mathbf{d}$ for corridor estimation can lead to potential suboptimality by invalidating regions with more consumption-friendly paths. To cope with the phenomena and not deviate from using solutions of the shortest path problem when determining corridors $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$, we propose two additional criteria functions. The first one is identical to the consumption cost function $\mathbf{c} : E \to \mathbb{R}$ defined in Section 2.1 and corridors $\mathcal{C}^{\epsilon}_{\mathbf{c}_{s,t}}$ induced by it represent the exact opposite to $\mathcal{C}^{\epsilon}_{\mathbf{d}_{s,t}}$. As the $P^{\mathbf{c}}_{s,t}$ is the most energy-efficient path from s to t, it passes through edges minimizing the overall consumption on the path between s and t.

The second additional criteria function $\boldsymbol{\omega} : E \to \mathbb{R}$ represents the compromise between $\mathbf{d}$ and $\mathbf{c}$ as it integrates the influence of both driving time

and the consumption on individual edges. It corresponds with the criteria function used for the computation of heuristic values later in the thesis inspired by the approach presented in [6]:

$$\boldsymbol{\omega}(e) = \mathbf{d}(e) + \frac{\mathbf{c}(e)}{\mathbf{cf}_{\max}}.$$

Here $\mathbf{d}(e)$ and $\mathbf{c}(e)$ are values of driving time and consumption functions for edges $e \in E$, and $\mathbf{cf}_{\max}$ stands for the fastest possible charging speed measured in percent per minute. Hence the second part of the function determines the time needed to be spent by charging an amount of energy consumed on edge e using the maximal possible constant charging speed $\mathbf{cf}_{\max}$. Therefore, the criteria function $\boldsymbol{\omega}$ represents a trade-off between $\mathbf{d}$ and $\mathbf{c}$ by converting an amount of consumed energy on edge to time.

Final Set of Corridors

The following set of corridors implied by functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$ together with corresponding shortest paths $P^{\mathbf{d}}_{s,t}$, $P^{\mathbf{c}}_{s,t}$ and $P^{\boldsymbol{\omega}}_{s,t}$ serves as an input for charging station selection technique presented in 3.2.2:

$$\mathcal{C}^{\epsilon}_{\{\mathbf{d},\mathbf{c},\boldsymbol{\omega}\}_{s,t}} = \{\mathcal{C}^{\epsilon}_{\mathbf{d}_{s,t}}, \mathcal{C}^{\epsilon}_{\mathbf{c}_{s,t}}, \mathcal{C}^{\epsilon}_{\boldsymbol{\omega}_{s,t}}\}.$$

By criteria functions $\mathbf{d}$, $\mathbf{c}$, $\boldsymbol{\omega}$ we estimate regions through which the final path $P_{s,t}$ can pass. However, the concept of general criteria functions $\mathbf{q}$ enables the introduction of even more exotic functions. It is possible to capture, for example, the amount and quality of POIs in the region, and this way let the planner lead the final path $P_{s,t}$ through some specific customizable regions. However, we do not expand this idea more in the theses and leave it as one of the concepts for future work.

3.2.2 Charging Stations Selection in Corridors

We define the mechanism of selection of charging stations in given corridors $\mathcal{C}^{\epsilon}_{\mathbf{d}_{s,t}}$, $\mathcal{C}^{\epsilon}_{\mathbf{c}_{s,t}}$ and $\mathcal{C}^{\epsilon}_{\boldsymbol{\omega}_{s,t}}$ regardless the criteria function hence the general function $\mathbf{q}$ is used in the following text. A heuristic choosing charging stations to take part in the auxiliary graph should take into account the probability of each charging station to be included in the final plan R defined in Section 2.2. We identify three main factors contributing to the probability:

- **Location** – determined by corridors $\mathcal{C}^{\epsilon}_{\mathbf{c}_{s,t}}$ and the distance from the path $P^{\mathbf{q}}_{s,t}$. Charging stations should be appropriately distributed along the path $P^{\mathbf{q}}_{s,t}$ to capture the potential need of charging.
- **Charging speed** – considering the assumption that faster charging stations represent a higher probability of taking part in the final plan R as they provide better time efficiency.

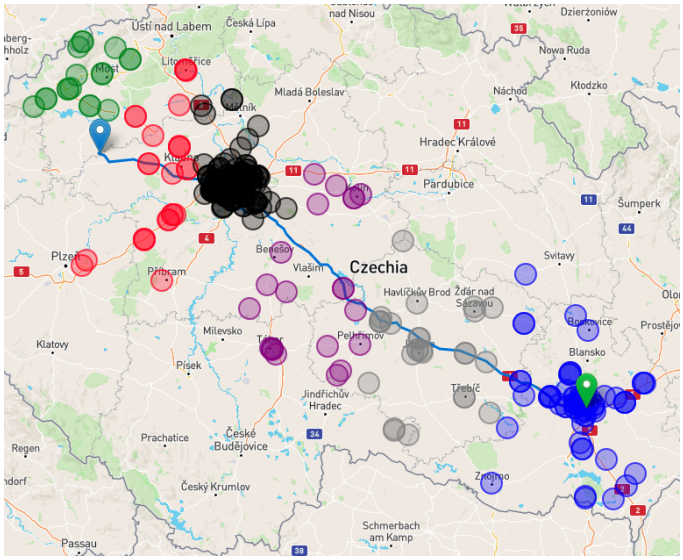


Figure 3.4: Clustered chargers from the $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ corridor to k clusters.

- **SoC context on the final path** – in the case of a high value of initial SoC β_s, there is a lower probability for the presence of a stop for charging at the beginning of the path.

In the first phase we extract all charging stations present in the corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ forming the set:

$$S^{\epsilon}_{\mathbf{q}_{s,t}} = \{v \in S \mid v \in \mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}\}.$$

Furthermore, we cluster the $S^{\epsilon}_{\mathbf{q}_{s,t}}$ by using a k-means algorithm firstly presented in [36], where:

$$k = \left\lceil \frac{\mathbf{L}(P^{\mathbf{q}}_{s,t})}{cl} \right\rceil.$$

The number of clusters k is retrieved by the division of the path $P^{\mathbf{q}}_{s,t}$ length $\mathbf{L}(P^{\mathbf{q}}_{s,t})$ measured in meters by the constant $cl \in \mathbb{N}$ which can be intuitively understood as the length of a cluster and its function is to secure sufficient granularity of clusters along the whole path. In our experiments presented in Chapter 4, we use the value of 50000 m (50 km). The main purpose of clustering is to evenly distribute chargers along the path and localize regions with a high density of chargers. An example of the intermediate result after the clustering procedure of all charging stations in the corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ is in Figure 3.4, where the color signifies the membership to the cluster. Black and blue clusters above regions of the biggest cities, Prague and Brno, are apparent with a higher density of charging stations.

The clustering process produces k clusters of charging stations $Cl^{\mathbf{q}}_i \subseteq S^{\epsilon}_{\mathbf{q}_{s,t}}$, $i \in \{1, ..., k\}$ and k corresponding centroids $ce^{\mathbf{q}}_i \in V$, $i \in \{1, ..., k\}$ (centroids do not necessary match nodes $v \in V$, hence we match the closest $v \in V$ to each centroid produced by k-means). As the key task of the charging station selection heuristic is to minimize the number of unnecessary

chargers included in the auxiliary graph and save the computational time of optimization algorithms, we introduce an upper bound $o \in \mathbb{N}$. It stands for the maximal number of charging stations selected for a single corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$. An influence of o on the overall quality of the plan is discussed in Chapter 4. Furthermore, we evenly split the upper bound and choose $\lceil \frac{o}{k} \rceil$ charging stations from each individual cluster $Cl^{\mathbf{q}}_i$. For this purpose, we sort chargers in all clusters by a preference $\mathbf{y} : V^m \times V \to \mathbb{R}_{\geq 0}$. $\mathbf{y}$ maps the shortest path $P^{\mathbf{q}}_{s,t}$ and a charging station $v \in Cl^{\mathbf{q}}_i$ to a value defined by:

$$\mathbf{y}(P^{\mathbf{q}}_{s,t}, v) = 2 \cdot \left(\frac{\mathbf{hv}_{\min}(P^{\mathbf{q}}_{s,t}, v)}{m\_s} \right) + \mathbf{cf}_v^{-1}(\beta_{m\_out}) - \mathbf{cf}_v^{-1}(\beta_{m\_in}).$$

The preference $\mathbf{y}$ is a heuristic estimation of the charger's probability of taking part in the final plan. The first part of the preference expresses an estimated driving time from the closest vertex $u \in P^{\mathbf{q}}_{s,t}$ to the charging station $v \in Cl^{\mathbf{q}}_i$ and back (i.e., coefficient 2 at the beginning). The haversine distance is used to estimate the distance of the closest vertex $u \in P^{\mathbf{q}}_{s,t}$ to v measured in meters and constant $m\_s \in \mathbb{N}$ stands for the value of mean speed measured in m/min. The value of m_s used in Section 4 is set to 70 km/h after conversion from m/min. The second part of the preference expresses the charging time needed to spend in $v \in Cl^{\mathbf{q}}_i$ by charging from $m\_in \in [0, 100]$ to $m\_out \in [0, 100]$. The values of m_in and m_out stand for the mean value of SoC when arriving at charging stations and the mean value of SoC when leaving them and they are again empirically assigned in Chapter 4 to values 5 % and 50 %. Such a criteria function prioritizes faster charging stations that are located nearby the path $P^{\mathbf{q}}_{s,t}$. Nevertheless, it can also express a trade-off between more distant charging stations providing very high speed.

Up to this moment, the proposed technique of choosing charging stations ensures a uniform distribution of charging stations along the path and prioritization of these with higher speed which corresponds with the first two factors discussed at the beginning of Subsection 3.2.2. To capture even the third one, the SoC context, we introduce an adaptation factor function $\mathbf{ad} : V \times V \times [0, 100] \to [0, 1]$ which maps the starting vertex $s \in P^{\mathbf{q}}_{s,t}$, cluster centroid $ce^{\mathbf{q}}_i$ and the initial SoC β_s to the factor value which adjusts a number of chosen charging stations from each cluster $Cl^{\mathbf{q}}_i$ corresponding to the centroid $ce^{\mathbf{q}}_i$ (i.e., we choose $\mathbf{ad}(s, ce^{\mathbf{q}}_i, \beta_s) \cdot \lceil \frac{o}{k} \rceil$ chargers from each cluster $Cl^{\mathbf{q}}_i$). The function is defined as follows:

$$\mathbf{ad}(s, ce^{\mathbf{q}}_i, \beta_s) := \begin{cases} 0.2 & \text{if } \beta_s - \beta_{v_i} \geq 80 \\ 0.4 & \text{if } \beta_s - \beta_{v_i} \geq 70 \\ 0.6 & \text{if } \beta_s - \beta_{v_i} \geq 60 \\ 0.8 & \text{if } \beta_s - \beta_{v_i} \geq 40 \\ 1 & \text{otherwise.} \end{cases}$$

Here β_{v_i} corresponds to the SoC at the vertex $v_i \in P^{\mathbf{q}}_{s,t}$ which has the shortest haversine distance to the cluster centroid $ce^{\mathbf{q}}_i$ i.e., the vertex used for

computation of $\mathbf{hv}_{\min}(P^{\mathbf{q}}_{s,t}, ce^{\mathbf{q}}_i)$. As a result of applying the value of the adaptation factor **ad** function, we take half of the first $\lceil \frac{o}{k} \rceil$ chargers from the cluster $Cl^{\mathbf{q}}_i$ if the SoC in the area captured by the cluster is greater than 80 %, three quarters in the case of the SoC is above 60 % and all charging stations otherwise.

The set of chargers which enters to the following procedure of the auxiliary graph construction arises as a unification of charging stations chosen from all clusters $Cl^{\mathbf{q}}_i$, $i \in \{1, \dots k\}$ constructed for corridors for all three criteria functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$. We denote this set by S_{aux} and it is at the same time the first output of the auxiliary graph construction procedure.

3.2.3 Time and Consumption Matrices Between Chargers with Multiple Criteria Functions

To speed up the construction of the auxiliary graph in the query time, we introduce a technique of precomputing time and consumption between charging stations which is essential for the final construction of an auxiliary graph. Since the EVRT-POI problem defined in Section 2.2 is an adaptation of the CSP problem, and we do not know SoC values on paths between charging stations no sooner than in the query phase, a general and optimality preserving approach to precompute these values uses algorithms solving the CSP which is the $\mathcal{NP}$-hard problem. This optimal approach also saves the whole set of non-dominated paths between chargers. The need to approach it this way is because the optimal constrained shortest path may differ regarding the initial SoC. Hence it is not enough to store a single path between two charging stations but the whole set of parallel paths. Such an approach is used, for example, in [19]. However, we decided to relax these conditions and replace the set of non-dominated constrained paths with the collection of three simple shortest paths, each optimizing one of the criteria functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$. The relaxation is furthermore discussed in Chapter 5.

For each two charging stations u, $v \in S$, $u \neq v$ we precompute shortest paths $P^{\mathbf{d}}_{u,v}$, $P^{\mathbf{c}}_{u,v}$ and $P^{\boldsymbol{\omega}}_{u,v}$ and save values $\mathbf{D}(P^{\mathbf{d}}_{u,v})$, $\mathbf{D}(P^{\mathbf{c}}_{u,v})$, $\mathbf{D}(P^{\boldsymbol{\omega}}_{u,v})$, $\mathbf{C}(P^{\mathbf{d}}_{u,v})$, $\mathbf{C}(P^{\mathbf{c}}_{u,v})$ and $\mathbf{C}(P^{\boldsymbol{\omega}}_{u,v})$. In the following text, we assume that there are edges $(u,v)_{\mathbf{q}}$ for all criteria functions $\mathbf{q} \in \{\mathbf{d}, \mathbf{c}, \boldsymbol{\omega}\}$ between each pair of charging stations u, $v \in S$ and we use functions $\mathbf{d}((u,v)_{\mathbf{q}}) = \mathbf{D}(P^{\mathbf{q}}_{u,v})$ and $\mathbf{c}((u,v)_{\mathbf{q}}) = \mathbf{C}(P^{\mathbf{q}}_{u,v})$ to retrieve the time and consumption along these edges. These values serve also as parameters for construction of three parallel edges between all chosen charging stations in S_{aux} in an auxiliary graph. Similarly, as discussed in Subsection 3.2.1, the purpose of these three parallel edges is to compensate for relaxations following from the use of unconstrained shortest path for precomputation of paths for constrained EVRT-POI problem.

3.2.4 Auxiliary Graph Construction

Considering the set of chosen charging stations S_{aux} produced by techniques presented in previous Subsections 3.2.1, 3.2.2 and 3.2.3, we define the full

auxiliary graph $G'_{\text{aux}} = (V'_{\text{aux}}, E'_{\text{aux}})$ for query Q, where $V'_{\text{aux}} = \{s, t\} \cup S_{\text{aux}}$ and E'_{aux} consists of two types of edges:

- $E'_{S_{\text{aux}}}$ – edges between chosen charging stations S_{aux}. There are three parallel edges $e_{\mathbf{d}}, e_{\mathbf{c}}, e_{\boldsymbol{\omega}}$ between each pair of charging stations. Each edge corresponds to one of criteria functions $\mathbf{d}$, $\mathbf{c}$, $\boldsymbol{\omega}$. All needed values to be able to determine driving time function $\mathbf{d}$ and consumption function $\mathbf{c}$ for edges $e_{\mathbf{q}} \in E'_{S_{\text{aux}}}$ in the same way as in the case of edges $e \in E$, are precomputed in time and consumption matrices described in the previous Subsection 3.2.3.
- $E'_{s,t}$ – edges from s to all charging stations in S_{aux}, edges from all charging stations in S_{aux} to t and one extra edge from s to t. To determine values of driving time function $\mathbf{d}$ and consumption function $\mathbf{c}$ for edges $e_{\mathbf{q}} \in E'_{s,t}$, shortest path queries for all three criteria $\mathbf{d}$, $\mathbf{c}$, $\boldsymbol{\omega}$ are performed during the auxiliary graph construction.

Parallel edges between each pair of charging stations $u, v \in S_{\text{aux}}$ are shown in Figure 3.1. Edges corresponding to distinct criteria functions are marked by different colors. To further prune the full auxiliary graph G'_{aux} and save the query time, we introduce the following pruning policy:

- Edge $e_{\mathbf{q}} \in E'_{\text{aux}}$, $e_{\mathbf{q}} = (u, v)_{\mathbf{q}}$ is pruned in the case $\mathbf{c}(e_{\mathbf{q}}) \geq 100$ i.e., if its consumption exceeds the maximal value of SoC and though it is not possible to traverse it without any charging stop.
- Edge $e_{\mathbf{q}} \in E'_{\text{aux}}$, $e_{\mathbf{q}} = (u, v)_{\mathbf{q}}$ is pruned in the case $\mathbf{d}((u, t)_{\mathbf{q}}) \leq \mathbf{d}((v, t)_{\mathbf{q}})$ and $\mathbf{c}((u, t)_{\mathbf{q}}) \leq \mathbf{c}((v, t)_{\mathbf{q}})$ i.e., if the edge from u to t takes shorter time and it consumes less energy. This policy aims to delete unnecessary edges returning back towards the start of the path s.

In the end we receive the auxiliary graph $G_{\text{aux}} = (V_{\text{aux}}, E_{\text{aux}})$ by applying the pruning policy described above to the full auxiliary graph G'_{aux}. By saving and using merely consumption values of edges $e_{\mathbf{q}}, \in E_{\text{aux}}$ we relax consumption constraints regarding regenerative breaking. For example, if there is a hill in the middle of the edge $(u, v)_{\mathbf{q}}$ from the vertex u, to the vertex v, the total consumption $\mathbf{c}((u, v)_{\mathbf{q}})$ may be lower than the value of the SoC needed to traverse the edge. This is caused by the fact that β_u has to be high enough to enable the EV to climb the hill. Still, as the EV reaches the summit, it starts to recuperate energy when going down the hill to its destination vertex v. This leads to a value of consumption on edge $(u, v)_{\mathbf{q}}$ which is lower than the value of SoC β_u at vertex u needed to traverse the edge. In the following text, we relax this phenomenon and discuss possible advantages, shortcomings, and distinct approaches in Chapter 5.

3.3 Consumption Model

The consumption of EVs is heavily affected by various distinct factors such as driving speed, air density, aerodynamic properties of EV, elevation pro-

file, the weight of the EV, and others. At the same time, the EV consumption directly affects SoC during the journey, and SoC is one of the critical components of the planning process for EVs as it dramatically affects the path and the overall plan, as discussed in Subsection 3.2.1. To model the EV consumption, we joined concepts introduced in [57, 1, 18] and elaborated our model, which enables us to compute consumption along route graph edges $e \in E$.

3.3.1 Power Estimation

In papers [57, 1], it is derived that the total force F acting on the EV in motion consists of the following components:

$$F = F_{\text{roll}} + F_{\text{grade}} + F_{\text{air}} + F_{\text{acc}}.$$

F_{roll} is the rolling resistance force between tires and the road, F_{grade} is the force resulting from differences in elevation, F_{air} represents the resistance of EV against air, and F_{acc} is the acceleration force. The power P measured in watts is computed by multiplication of force F by the driving velocity $v \in \mathbb{R}_{\geq 0}$. For our purposes, we assume a constant vehicle velocity on edges $e \in E$, which results in zero acceleration value and hence force F_{acc} does not contribute to the overall value of F. Individual force components are computed as follows:

- $F_{\text{roll}} = m \cdot g \cdot f_r$ – where m is the total mass of the EV, g is the gravitational acceleration constant, and f_r stands for the tire rolling resistance affected by the type of the tire and the road quality.
- $F_{\text{grade}} = m \cdot g \cdot i$ – where i is a road grade measured in radians implied by elevation changes along the path.
- $F_{\text{air}} = \frac{1}{2} \cdot \rho_{\text{air}} \cdot C_d \cdot A_f \cdot v^2$ – where ρ_{air} is an air density directly affected by the temperature, C_d is the aerodynamic drag coefficient expressing the aerodynamics of the EV shape, A_f captures the frontal area measured in m^2 and v stands for the EV velocity.

The final equation for the power measured in watts is:

$$P = v \cdot (m \cdot g \cdot f_r + m \cdot g \cdot i + \frac{1}{2} \cdot \rho_{\text{air}} \cdot C_d \cdot A_p \cdot v^2) \text{ [W]}.$$

3.3.2 Regenerative Braking Efficiency

When EV decelerates, an amount of energy is released, and some part of it may be stored back in the battery. Such a process is called regenerative braking or recuperation. Fiori et al. [18] state that there is an exponential relationship between the efficiency factor of regenerative braking η and negative acceleration of the vehicle $a^{(-)}$. We estimate this acceleration as $a^{(-)} = g \cdot (f_r + i)$ as it is caused by elevation changes when going down the hill

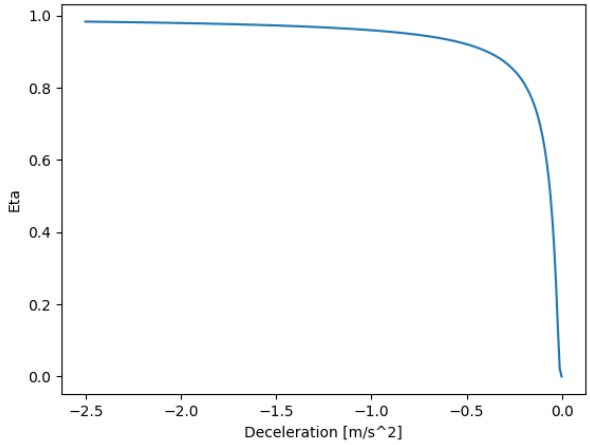


Figure 3.5: Dependency of regenerative braking efficiency on deceleration $a^{(-)}$.

and the driver's effort to decelerate to keep a constant velocity. The final equation for η published in [18] is:

$$\eta := \begin{cases} 1 & \text{if } a^{(-)} \geq 0 \\ \left[e^{\left(\frac{\alpha}{|a^{(-)}|} \right)} \right]^{-1} & \text{otherwise.} \end{cases}$$

$\alpha \approx 0.0411$ stands for the hyperparameter received from the least-squares optimization method as Fiori et al. estimate the final parameters of the exponential function empirically from reported data on regenerative braking efficiency. As it is shown in Figure 3.5, where on the x axis there is a deceleration $a^{(-)}$ measured in $\mathrm{m/s^2}$ and on the y axis there are values of η, the efficiency for deceleration values between 0 and -0.5 $\mathrm{m/s^2}$ rapidly rises from zero up to values exceeds 0.9. As soon as the value of deceleration surpasses the value of -0.5 $\mathrm{m/s^2}$, the efficiency η remains in high values aiming towards one.

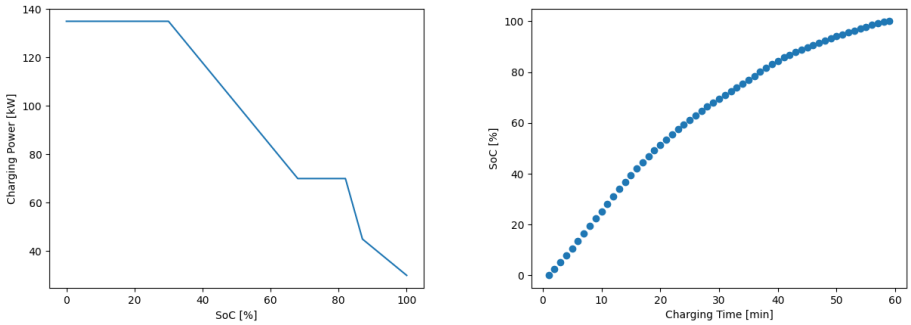
3.3.3 Consumption on Individual Edges

Considering all components defined in previous Subsections 3.3.1 and 3.3.2, we define the following implementation of the consumption function $\mathbf{c}_{\mathrm{kWh}} : E \to \mathbb{R}$:

$$\mathbf{c}_{\mathrm{kWh}}(e) := \frac{(\eta \cdot P + P_{aux})}{1000} \cdot \frac{\mathbf{d}(e)}{60} \text{ [kWh]}.$$

P_{aux} stands for auxiliary power needed by auxiliary electric devices, 1000 is a conversion constant from W to kW, and 60 is a constant used for conversion from minutes to hours. Considering the EV battery capacity $M \in \mathbb{R}$ measured in kWh, we can introduce the equation for the consumption function $\mathbf{c}$ measured in percent:

$$\mathbf{c}(e) = \frac{\mathbf{c}_{\mathrm{kWh}}(e)}{M} \cdot 100 \text{ [\%]}.$$



(a) : Charging curve **cc** of Skoda Enyaq iV 80 [15].

(b) : Partial charging function $\mathbf{cf}_v$ defined in breakpoints $k \cdot ls$.

Figure 3.6: Examples of charging curve **cc** and partial charging function $\mathbf{cf}_v$.

3.4 Charging Model

To model the charging process, we introduce a mechanism enabling the conversion of charging curves published by EV users on web pages such as [15] into a piecewise linear charging functions $\mathbf{cf}_v$ and $\mathbf{cf}_v^{-1}$ introduced in Section 2.1. An example of such a charging curve is in Figure 3.6a, which shows the charging curve of Skoda Enyaq iV 80. On the x axis, there are values of SoC measured in percent, and on the y axis, there is a charging power in kW, which is what the EV is capable of receiving at the corresponding SoC value. As it is clear from Figure 3.6a, the power rapidly decreases with higher values of SoC, which results in the fact that the suitable interval for charging in terms of speed is approximately between 0 and 50 % in the case of Enyaq.

3.4.1 Conversion of Charging Curve to Charging Function

We define charging curves similar to the one in Figure 3.6a formally as a piecewise linear charging curve functions $\mathbf{cc} : [0, 100] \to \mathbb{R}_{\geq 0}$ mapping SoC values measured in percent to the power P_{in} measured in kW. For modeling charging function $\mathbf{cf}_v$, we identify the vector Br of $n \in \mathbb{N}$ charging breakpoints $br \in [0, 100] \times \mathbb{R}$. These breakpoints correspond to values, where the charging curve function **cc** switches from one linear function to another. We add borders of the function **cc** to the beginning and at the end of Br as well. Each breakpoint is represented by a tuple of SoC β_i measured in percent and corresponding power P_{in_i} measured in kW, for all $i \in \{1, \ldots, n\}$:

$$Br = [(\beta_1, P_{\text{in}_1}), \ldots, (\beta_n, P_{\text{in}_n})].$$

Since we model charging functions $\mathbf{cf}_v$ as piecewise linear functions, we introduce a linearity interval size $ls \in \mathbb{R}_{\geq 0}$ which represents the length of a time interval between breakpoints of $\mathbf{cf}_v$ measured in hours which is assumed as

constant. Considering $\mathbf{cc}$, Br and ls we can now define the charging function $\mathbf{cf}_v : \mathbb{R}_{\geq 0} \to [0, 100]$ for values of charging time being equal to k multiples of ls, where $k \in \{0, 1, 2, \dots\}$ i.e., for values $t \in \{0, 1 \cdot ls, 2 \cdot ls, \dots\}$, recursively as follows:

$$\mathbf{cf}_v(k \cdot ls) := \begin{cases} 0 & \text{if } k = 0 \\ \mathbf{cf}_v((k-1) \cdot ls) \dfrac{100}{M} & \\ \quad + ls \cdot \mathbf{cc}(\mathbf{cf}_v((k-1) \cdot ls)) & \text{if } \mathbf{cf}_v((k-1) \cdot ls) < 100 \\ 100 & \text{otherwise.} \end{cases}$$

The fundamental principle of the computation is in the application of the known equation for the calculation of energy E from power P measured in kilowatts and time t measured in hours:

$$E = P \cdot t \text{ [kWh]}.$$

Initially, the value of charged energy for $k = 0$, i.e., after charging time $t = k \cdot ls = 0 \cdot ls = 0$ when charging from zero percent is equal to zero percent. The second case in the equation applies if the value of SoC charged after time $(k-1) \cdot ls$ (which is already known) is below 100 %. In this case, the value of $\mathbf{cf}_v(k \cdot ls)$ is computed as the sum of an amount of energy charged after time $(k-1) \cdot ls$ (already known) and an amount of energy charged during the interval of ls hours using the charging power corresponding to the value of SoC at the previous breakpoint which is equal to $\mathbf{cf}_v((k-1) \cdot ls)$. This is the place of application of the equation for computation of an energy from power P and time t, where power $P = \mathbf{cc}(\mathbf{cf}_v((k-1) \cdot ls))$ and time $t = ls$. The fraction $\frac{100}{M}$ in the equation provides the conversion from kWh to percent. In the last case when neither of two conditions in first two cases is fulfilled the value of $\mathbf{cf}_v$ is set as a constant to 100 %. By applying previously described recursive equation gradually to values $t \in [0, 1 \cdot ls, 2 \cdot ls, \dots k \cdot ls]$, where $k \cdot ls$ is the first value of time for which $\mathbf{cf}_v(k \cdot ls) \geq 100$, we receive amounts of energy $[0, \beta_1, \dots, \beta_k]$ measured in percent charged in v corresponding to individual charging times t. Such values are shown in Figure 3.6b, where on the x axis, there is a charging time measured in minutes and on the y axis there is the SoC measured in percent. The value of ls in Figure 3.6b is equal to one minute, and the partial charging function $\mathbf{cf}_v$ was computed using the charging curve function $\mathbf{cc}$ shown in Figure 3.6a.

Values of $\mathbf{cf}_v$ for $t \notin \{0, 1 \cdot ls, 2 \cdot ls, \dots\}$ are retrieved by linear interpolation of corresponding breakpoint values satisfying $(k-1) \cdot ls < t < k \cdot ls$. Function $\mathbf{cf}_v^{-1}$ mapping the target SoC to the needed charging time is received by a trivial inversion of $\mathbf{cf}_v$. Examples of $\mathbf{cf}_v$ and $\mathbf{cf}_v^{-1}$ produced by the mechanism described above are shown in Figures 2.2a and 2.2b. By the time interval ls, it is possible to model granularity of $\mathbf{cf}_v$ functions. In our experiments presented in Chapter 4, we use the value of one minute, i.e., $ls = \frac{1}{60}$ hours. For the purpose of saving runtime of optimization algorithms, it is possible to precompute breakpoint values of charging functions $\mathbf{cf}_v$ for all different types (varying charging station powers) of charging stations $v \in S$.

3.4.2 Computation of Charging Time

Since we define charging functions $\mathbf{cf}_v$ for charging from the SoC of zero percent, an additional mechanism is needed to be introduced for determining the final SoC β_{out} when charging for time t from the initial SoC β_{in}. For this purpose, we use inverse charging functions $\mathbf{cf}_v^{-1}$ mapping a value of SoC β_{out} measured in percent to the time needed to spend charging at a charging station $v \in S$ when charging from zero. The value of β_{out} when charging for t minutes at the charging station $v \in S$ from the initial SoC β_{in} is retrieved by:

$$\beta_{\text{out}} = \mathbf{cf}_v(t + \mathbf{cf}_v^{-1}(\beta_{\text{in}})).$$

Similar mechanism was used in [6] as well.

3.5 Proposed Optimization Algorithms

We propose two approaches to solving the EVRT-POI problem defined in Section 2.2. Both of them operate with the auxiliary graph G_{aux} and consider consumption and charging models as defined in previous Sections 3.3 and 3.4. First, we introduce an Integer Linear Programming (ILP) model with relaxations on POI scheduling which provides a straightforward solution for EV routing problems with integrated charging stations. It serves as a baseline algorithm for the final solution based on the CFP algorithm proposed in [6]. Our CFP-POI solves the EVRT-POI problem to its full extent. Besides the heuristic based on criteria function $\boldsymbol{\omega}$, we introduce a new heuristic approach based on precomputing plan duration between specific locations to accelerate the algorithm during the query phase.

The initial part of solving the EVRT-POI problem is shared between both approaches. It consists of processing query Q by applying techniques described in Section 3.2, resulting in generation of set of chosen charging stations S_{aux}, corresponding set of POIs located nearby charging stations L_{aux} and the auxiliary graph G_{aux}. These three components are inserted into an updated query $Q_{\text{aux}} = (G_{\text{aux}}, s, t, \beta_s, \beta_t, S_{\text{aux}}, L_{\text{aux}}, d_{\text{POI}}, f_{\text{POI}})$ which serves as an input for both algorithms.

3.5.1 Integer Linear Programming Model

For the sake of simplicity, in the ILP model presented below, we do not consider scheduling of POIs. Therefore, it operates with limited queries $Q_{\text{aux}} = (G_{\text{aux}}, s, t, \beta_s, \beta_t, S_{\text{aux}})$ without parameters for POI scheduling. Nevertheless, POI scheduling can be incorporated into the proposed model as well. Furthermore, we prune the auxiliary graph G_{aux} and assume merely edges $e_{\mathbf{d}} \in E_{\text{aux}}$ optimizing the driving time criteria function $\mathbf{d}$.

The objective of the model is to find the plan $R_{\text{aux}} = (P_{s,t}^{\text{aux}}, T_{P_{s,t}^{\text{aux}}})$, where $P_{s,t}^{\text{aux}}$ is the path minimizing the overall driving and charging time in the given auxiliary graph G_{aux} and $T_{P_{s,t}^{\text{aux}}}$ is a charging time profile of the path $P_{s,t}^{\text{aux}}$ determining charging times in individual vertices on the path. Since we consider

just time optimizing edges $e_{\mathbf{d}} \in E_{\text{aux}}$, we do not specify the criteria function for individual edges throughout the definition and description of the ILP model. The definition of the model follows:

$$\min \sum_{(u,v)\in E_{\text{aux}}} \mathbf{d}(u,v) \cdot x_{(u,v)} + \sum_{v \in S_{\text{aux}}} ct_v + \sum_{v \in S_{\text{aux}}} z_v \cdot Ex_v$$

subject to:

$$\begin{array}{lllll}
(\text{A1}): & \forall e \in E_{\text{aux}} & x_e & \in \{0,1\} & \\
(\text{A2}): & \forall v \in S_{\text{aux}} & z_v & \in \{0,1\} & \\
(\text{A3}): & \forall v \in S_{\text{aux}} & ct_v & \geq 0 & ct_v \in \mathbb{R} \\
(\text{A4}): & \forall v \in V_{\text{aux}} & s_i_v & \geq \boldsymbol{\beta}_{\min}(v) & s_i_v \in \mathbb{R} \\
(\text{A5}): & \forall v \in V_{\text{aux}} & s_o_v & \geq \boldsymbol{\beta}_{\min}(v) & s_o_v \in \mathbb{R} \\
(\text{A6}): & \forall v \in S_{\text{aux}} & q_v & \geq 0 & q_v \in \mathbb{R} \\
(\text{A7}): & \forall v \in S_{\text{aux}} & \tau_v & \geq 0 & \tau_v \in \mathbb{R}
\end{array}$$

$$\begin{array}{lll}
(\text{B1}): & \sum_{u\in V_{\text{aux}}} x_{(s,u)} = 1 & \\
(\text{B2}): \forall v \in S_{\text{aux}} & \sum_{u\in V_{\text{aux}}} x_{(u,v)} = \sum_{w\in V_{\text{aux}}} x_{(v,w)} & \\
(\text{B3}): \forall v \in S_{\text{aux}} & \sum_{u\in V_{\text{aux}}} x_{(u,v)} = z_v &
\end{array}$$

$$\begin{array}{lll}
(\text{C1}): & s_o_s = \beta_s & \\
(\text{C2}): & s_i_t \geq \beta_t & \\
(\text{C3}): \forall (u,v) \in E_{\text{aux}} & x_{(u,v)} = 1 \implies s_i_v = s_o_u - \mathbf{c}(u,v) & \\
(\text{C4}): \forall v \in S_{\text{aux}} & q_v = \mathbf{cf}_v^{-1}(s_i_v) & \\
(\text{C5}): \forall v \in S_{\text{aux}} & \tau_v = q_v + ct_v & \\
(\text{C6}): \forall v \in S_{\text{aux}} & s_o_v = \mathbf{cf}_v(\tau_v) &
\end{array}$$

We start the ILP model's description with constraints divided into three groups regarding their meaning.

Variable Types and Value Constraints

The first set of constraints A1–A7 defines all used variables:

- (A1) – defines binary variables x_e for all edges $e \in E_{\text{aux}}$. If the variable $x_e = 1$, it means that the edge e takes part in the path $P^{\text{aux}}_{s,t}$. It is not used otherwise.
- (A2) – defines continuous variables z_v for all vertices $v \in S_{\text{aux}}$. If the variable $z_v = 1$, it means that there is a charging action scheduled in v. No charging in v is scheduled otherwise.
- (A3) – defines continuous variables ct_v for all vertices $v \in S_{\text{aux}}$. Variables ct_v are greater or equal to zero and they stand for the value of scheduled charging time in vertices $v \in S_{\text{aux}}$.

- (A4) – defines continuous variables s_i_v for all vertices $v \in V_{\text{aux}}$. Variables s_i_v are greater or equal to the minimal SoC $\boldsymbol{\beta}_{\min}(v)$ at corresponding vertices v and they stand for the value of SoC measured in percent when entering vertices $v \in V_{\text{aux}}$.
- (A5) – defines continuous variables s_o_v for all vertices $v \in V_{\text{aux}}$. Variables s_o_v are greater or equal to the minimal SoC $\boldsymbol{\beta}_{\min}(v)$ at corresponding vertices v and they stand for the value of SoC measured in percent when leaving vertices $v \in V_{\text{aux}}$.
- (A6) – defines continuous variables q_v for all vertices $v \in S_{\text{aux}}$. Variables q_v are greater or equal to zero and they stand for the value of charging time needed to spend by charging at the charging station v to charge to some value of SoC when charging from zero. It is assigned by the constraint (C4).
- (A7) – defines continuous variables τ_v for all vertices $v \in S_{\text{aux}}$. Variables τ_v are greater or equal to zero and they stand for the value of charging time needed to spend by charging at the charging station v to charge to some value of SoC when charging from zero. It is assigned in the constraint (C5). The difference between variables τ_v and q_v is further explained by the description of constraints (C4) and (C5).

Constraints Enforcing the Path Continuity

- (B1) – enforces leaving from the initial vertex s by exactly one edge.
- (B2) – ensures that whenever the EV enters some vertex $v \in S_{\text{aux}}$ from vertex $u \in V_{\text{aux}}$, there has to exist some vertex $w \in V_{\text{aux}}$ it continues to on the path.
- (B3) – enforces charging whenever EV visits the vertex $v \in S_{\text{aux}}$.

Consumption and Charging Constraints

- (C1) – ensures that the leaving SoC s_o_s of the initial vertex s is equal to the given value of the initial SoC β_s for the query Q.
- (C2) – ensures that the entering SoC s_i_t of the destination vertex t is greater or equal to the given value of the destination SoC β_t for the query Q.
- (C3) – defines that whenever the binary variable $x_{(u,v)}$ is equal to 1, the entering SoC s_i_v at the vertex v is equal to the output SoC s_o_u in the previous vertex u minus the consumption $\mathbf{c}(u, v)$ on the edge (u, v).
- (C4) – q_v is a helping variable for application of the mechanism of determining the charging time to some SoC s_o_v from non zero s_i_v described in Subsection 3.4.2. q_v is assigned to the value of charging

time needed to charge from zero to some specific value of entering SoC s_i_v.

- (C5) – τ_v is a helping variable for application of the mechanism of determining the charging time to some SoC s_o_v from non zero s_i_v described in Subsection 3.4.2. τ_v is assigned to the value of charging time needed to charge from zero to some specific value of leaving SoC s_i_v which is equal to q_v plus the value of variable ct_v which stands for the real charging time spend by charging in v by charging from s_i_v to s_o_v.
- (C6) – defines that the SoC s_o_v when leaving vertex $v \in S_{\text{aux}}$ is equal to the value of the charging function $\mathbf{cf}_v$ after charging for time τ_v defined in the previous constraint (C5).

Finally, the criteria function consists of the first sum which stands for the driving time needed to traverse the resulting path $P^{\text{aux}}_{s,t}$. The sum consists only of these values of $\mathbf{d}(u, v)$ which take part in the $P^{\text{aux}}_{s,t}$ and all the other values are multiplied by zero. The following two sums model the time spent by charging. The first one sums values of all charging time variables ct_v, which results in the total charging time. Constants $Ex_v \in \mathbb{R}_{\geq 0}$ defined for all charging stations $v \in S_{\text{aux}}$ stand for an extra penalty used for each charging action. They model manipulation and possibly waiting times in queues in front of charging stations. Therefore, the last sum stands for the total manipulation and waiting time.

It is important to note, that the ILP model as defined above, does not precisely satisfy the definition of the Integer Linear Programming since constraints (C3), (C4) and (C6) are not linear. However, the majority of modern solvers can linearize such constraints and transform the problem into the raw ILP model.

By solving the ILP model, we receive the assignment of variables $x_{(u,v)}$, $\forall (u, v) \in E_{\text{aux}}$, which enables the reconstruction of the path $P^{\text{aux}}_{s,t} = [s = v_1, \ldots, t = v_m]$. The charging profile $T_{P^{\text{aux}}_{s,t}}$ is reconstructed from the assignment of variables ct_v for vertices v in the path $P^{\text{aux}}_{s,t}$. By joining the path $P^{\text{aux}}_{s,t}$ and the charging profile $T_{P^{\text{aux}}_{s,t}}$ we receive the plan $R_{\text{aux}} = (P^{\text{aux}}_{s,t}, T_{P^{\text{aux}}_{s,t}})$. To further convert the R_{aux} to the plan R defined as the solution of the EVRT-POI problem, the shortest path queries in original route graph G optimizing the criteria function $\mathbf{d}$ are computed between all edges (v_i, v_{i+1}), $i \in \{1, \ldots, m-1\}$ for $v_i \in P^{\text{aux}}_{s,t}$. We compute merely shortest path optimizing the driving time criteria function, because we restricted the ILP model to work only with a single edge $e \in E_{\text{aux}}$ between two vertices $v \in V_{\text{aux}}$. The shortest paths are then connected in the corresponding order forming the final path $P_{s,t}$. Charging profile $T_{P_{s,t}}$ is formed by setting values corresponding with vertices $v \in P^{\text{aux}}_{s,t}$ from charging profile $T_{P^{\text{aux}}_{s,t}}$ to corresponding values in $T_{P_{s,t}}$. All the other values are considered zero. As we do not assume POI scheduling in the ILP model POI schedule $Z_{P_{s,t}} = \{\}$ is empty and we have the final plan for the query Q:

$$R = (P_{s,t}, T_{P_{s,t}}, \{\}).$$

3.5.2 Charging Function Propagating Algorithm with Points of Interest

The CFP-POI is the exact algorithm based on the CFP algorithm proposed in [6], which is an adaptation of the bicriteria variant of the Dijkstra's algorithm introduced in [37]. The CFP-POI adopts most critical features from the CFP, introduces several simplifications resulting from characteristics the auxiliary graph, and extends it by introducing additional mechanisms used for scheduling POIs.

Similarly as in the case of the ILP model defined in Subsection 3.5.1, the CFP-POI receives the query $Q = (G, s, t, \beta_s, \beta_t, S, L, d_{\text{POI}}, f_{\text{POI}})$ as an input. Techniques described in Section 3.2 are applied, the auxiliary graph G_{aux} is constructed, and the corresponding query Q_{aux} enters the CFP-POI algorithm:

$$Q_{\text{aux}} = (G_{\text{aux}}, s, t, \beta_s, \beta_t, S_{\text{aux}}, L_{\text{aux}}, d_{\text{POI}}, f_{\text{POI}}).$$

CFP-POI algorithm maintains a priority queue of labels ℓ, defined in Subsection 3.5.2, that capture the state in the EVRT-POI search space. In each step of the algorithm, one label is extracted from the priority queue and checked for satisfying termination conditions. Then it is further expanded by producing new labels originating from the extracted one. These labels are inserted back into the priority queue and the algorithm continues until the first extracted label ℓ meets all termination conditions:

1. The label ℓ is located in the terminal vertex t.
2. There is no occurrence of running out of energy on the path induced by the label ℓ.
3. There is no occurrence of violation of constraints regarding the scheduling of POIs.

Feasibility Function

For the purpose of modeling consumption constraints, we define the feasibility function $\mathbf{b}_{(u,v)_\mathbf{q}} : [0, 100] \to [0, 100] \cup \{-\infty\}$ for each edge $(u, v)_\mathbf{q} \in E_{\text{aux}}$ in the auxiliary graph G_{aux}, where the $\mathbf{q}$ stands for the criteria function used for the shortest path query during the auxiliary graph construction. The feasibility function $\mathbf{b}_{(u,v)_\mathbf{q}}$ maps the value of the SoC β_u at the vertex u to the SoC value β_v at the vertex v. It is defined as follows:

$$\mathbf{b}_{(u,v)_\mathbf{q}}(\beta_u) := \begin{cases} -\infty & \text{if } \beta_u < \mathbf{c}((u,v)_\mathbf{q}) + \boldsymbol{\beta}_{\min}(v) \\ \min\{\beta_u - \mathbf{c}((u,v)_\mathbf{q}), 100\} & \text{otherwise.} \end{cases}$$

In the case β_u is not sufficient for the traversal of the edge $(u, v)_\mathbf{q}$, the feasibility function returns $-\infty$, which stands for the infeasible β_u. Otherwise, the value of $\mathbf{b}_{(u,v)_\mathbf{q}}(\beta_u)$ is equal to the SoC value β_v at the vertex v after the traversal of the edge $(u, v)_\mathbf{q}$, which is computed as the difference between β_u and the consumption of the edge $\mathbf{c}((u, v)_\mathbf{q})$. In the case when such value surpasses 100 %, the maximal SoC is returned.

CFP-POI Labels

The label ℓ in the CFP-POI algorithm consists of the following components:

- $v \in V_{\text{aux}}$ – a current vertex characterizing the label ℓ.
- $u \in S_{\text{aux}}$ – the last visited charging station on the path $P^{\text{aux}}_{s,v}$ from the initial vertex s to the current vertex v. The previous charging station u may be undefined in the case when the vertex characterizing the label ℓ_{par} from which the current label ℓ originates, is the starting vertex s and s is not a charging station. Otherwise, u is the vertex characterizing the previous label ℓ_{par} since the auxiliary graph G_{aux} consists merely from the starting vertex s, the destination vertex t and all the other vertices are charging stations.
- $\beta_u \in [0, 100]$ – a value of the SoC when arriving to the previous charging station at the vertex u.
- $\mathcal{T}_{\text{trip}} \in \mathbb{R}_{\geq 0}$ – trip time (driving, charging, POI stops) from the initial vertex s to the current vertex v without the charging time in the previous charging station at the vertex u but including $Ex_u \in \mathbf{R}_{\geq 0}$ which stands for the extra penalty modeling waiting and manipulation times at charging stations $u \in S_{\text{aux}}$ similar as in the case of the criteria function in the ILP model defined in Subsection 3.5.1.
- $\mathbf{q} \in \{\mathbf{d}, \mathbf{c}, \boldsymbol{\omega}\}$ – a criteria function $\mathbf{q}$ used for the computation of the time and consumption of the edge $(u, v)_{\mathbf{q}}$.
- $key \in \mathbb{R}_{\geq 0}$ – the minimal feasible trip time of the path $P^{\text{aux}}_{s,v}$ from the initial vertex s to the current vertex v regarding the charging at the previous charging station u:

$$key := \begin{cases} \mathcal{T}_{\text{trip}} & \text{if } \beta_u \geq \mathbf{c}((u,v)_{\mathbf{q}}) + \boldsymbol{\beta}_{\min}(v) \\ \mathcal{T}_{\text{trip}} + \mathcal{T}_{\text{key}} & \text{otherwise.} \end{cases}$$

The value of the key is equal to the time of the trip $\mathcal{T}_{\text{trip}}$ from the initial vertex s to the current vertex v if the value of the SoC β_u is high enough to traverse the edge $(u, v)_{\mathbf{q}}$ respecting consumption constraints. Otherwise the key consists of the sum of the trip time $\mathcal{T}_{\text{trip}}$ and $\mathcal{T}_{\text{key}}$ which stands for the minimal charging time at u to reach v respecting consumption constraints and it is computed as follows:

$$\mathcal{T}_{\text{key}} = \mathbf{cf}_u^{-1}(\mathbf{c}((u,v)_{\mathbf{q}}) + \boldsymbol{\beta}_{\min}(v) - \beta_u) - \mathbf{cf}_u^{-1}(\beta_u).$$

The first part of the equation stands for the charging time at the charging station u needed to charge from zero to the value of energy consumed by traversing the edge $(u, v)_{\mathbf{q}}$ augmented by the minimal SoC $\boldsymbol{\beta}_{\min}(v)$ at the vertex v and decreased by the SoC β_u at the vertex u. The second part stands for the charging time at the charging station in the vertex

u from zero to the value of SoC β_u at the vertex u. The subtraction of these two parts results in the minimal charging time needed to spend at the charging station at the vertex u to reach the vertex v and respect consumption constraints.

- $x_{\text{POI}} \in \{0, 1\}$ – a binary value determining whether POI stop is scheduled nearby the charging station at the vertex u_{parent} in the label ℓ_{par} defined below.
- $t_{\text{POI}} \in \mathbb{R}_{\geq 0}$ – a timestamp of the previously scheduled and finished stop in a POI measured in minutes. Timestamps are computed relative to the time zero, which is the time of the departure from the initial vertex s.
- ℓ_{par} – a pointer to the parent label the current label ℓ originates from.

The complete label can be written as a tuple:

$$\ell = (v, u, \beta_u, \mathcal{T}_{\text{trip}}, \mathbf{q}, key, x_{\text{POI}}, t_{\text{POI}}, \ell_{\text{par}}).$$

SoC Function

We define the SoC function $f\langle\ell\rangle$ for each label ℓ. It maps time $\mathcal{T} \in \mathbb{R}_{\geq 0}$ to the SoC β_v at the vertex v corresponding to the label ℓ. Depending on the input time $\mathcal{T}$ it adapts the charging time $\mathcal{T}_{\text{charge}}$ in the previous charging station u of the label and determines the SoC in v. The function is defined as follows:

$$f\langle\ell\rangle(\mathcal{T}) := \begin{cases} \mathbf{b}_{(u,v)_{\mathbf{q}}}(\mathbf{cf}_u(\mathbf{cf}_u^{-1}(\beta_u) + \mathcal{T} - \mathcal{T}_{trip})) & \text{if } \mathcal{T} \geq \mathcal{T}_{trip} \\ -\infty & \text{otherwise.} \end{cases}$$

In the case when given time $\mathcal{T}$ is greater than the current trip time $\mathcal{T}_{\text{trip}}$, i.e., induced charging time at the previous charging station u defined as $\mathcal{T}_{\text{charge}} = \mathcal{T} - \mathcal{T}_{\text{trip}}$ is greater or equal to zero, the SoC at v is equal to the value of the feasibility function $\mathbf{b}_{(u,v)_{\mathbf{q}}}$ applied to the SoC when leaving u after charging $\mathcal{T} - \mathcal{T}_{trip}$ time. Otherwise, the value $\mathcal{T}$ is infeasible for the label ℓ. An example of the SoC function is in Figure 3.7, where on the x axis, there is the time τ measured in minutes and on the y axis, there is the value of SoC when arriving to the vertex v corresponding to the label ℓ. The red part of the function stands for the $-\infty$, which is caused by infeasible values of τ.

The Dominance of Labels

The dominance for two labels ℓ and ℓ' is defined only in the case both labels share the same current vertex v. Label ℓ dominates the label ℓ' if the following holds for all $\mathcal{T} \in \mathbb{R}_{\geq 0}$:

$$f\langle\ell\rangle(\mathcal{T}) \geq f\langle\ell'\rangle(\mathcal{T}).$$

In words, the label ℓ dominates the label ℓ' if it can not happen that both labels share the same final trip time $\mathcal{T}$ and SoC in the second label ℓ' induced

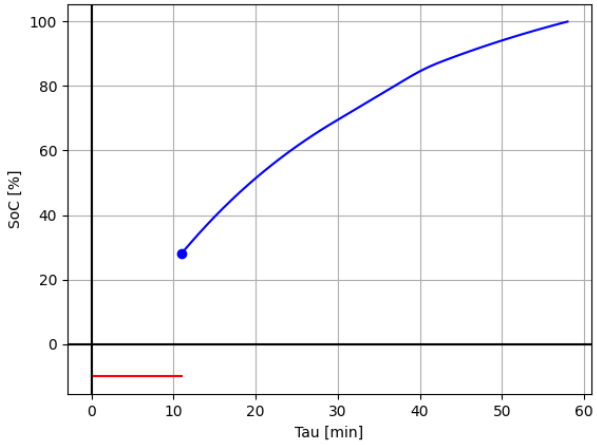


Figure 3.7: Example of the SoC function $f\langle\ell\rangle(\mathcal{T})$.

by the charging in the previous charging station u' would surpass the SoC in the label ℓ induced by the charging in the previous charging station u. Dominance checks for two labels ℓ and ℓ' are done by comparison of SoC values induced by breakpoints of corresponding charging functions $\mathbf{cf}_u$ and $\mathbf{cf}_{u'}$ since we consider piecewise linear functions.

CFP-POI Algorithm Description

The CFP-POI algorithm operates with the priority queue $\mathcal{Q}$ containing pairs of vertices $v \in V_{\text{aux}}$ and values of label parameters key which serve to determine the priority in the queue. The lower the value of the key is, the higher the priority of the corresponding pair in the $\mathcal{Q}$. This corresponds to the searching of the plan with the minimal feasible driving, charging, and POI stops time. In the succeeding text, we use the following notation for these pairs:

$$\jmath = (v, key).$$

Furthermore, the CFP-POI algorithm maintains sets $\mathcal{L}_{\text{set}}(v), \forall v \in V_{\text{aux}}$ containing labels already extracted from the priority queue $\mathcal{Q}$ during the process of the algorithm and sets $\mathcal{L}_{\text{uns}}(v), \forall v \in V_{\text{aux}}$ containing labels which have not been extracted yet. These two sets are defined individually for each vertex $v \in V_{\text{aux}}$ and hence we write $\mathcal{L}_{\text{set}}(v)$ and $\mathcal{L}_{\text{uns}}(v)$ to determine the specific set corresponding to the vertex $v \in V_{\text{aux}}$. All sets $\mathcal{L}_{\text{uns}}(v)$ are organized as priority queues with the parameter key of individual labels $\ell \in \mathcal{L}_{\text{uns}}(v)$ determining the priority. To distinguish between the label ℓ, $\ell' \in \mathcal{L}_{\text{uns}}(v)$ in the case of equality of key and key', the value of the SoC β_{key}, which corresponds to the SoC at the vertex v after charging for the minimal time $\mathcal{T}_{\text{key}}$ at the previous charging station, is used. The same policy for breaking ties is applied for the priority queue $\mathcal{Q}$. During the whole process of the algorithm, it applies that the pair $\jmath = (v, key)$ which is on the top of the priority queue

$\mathcal{Q}$ corresponds to the label ℓ on the top of the priority queue $\mathcal{L}_{\text{uns}}(v)$.

In each step of the CFP-POI algorithm, one label ℓ is extracted from $\mathcal{Q}$ and further expanded by producing new labels induced by charging at the previous charging station u in the label ℓ or by driving to some neighboring label ℓ' along with one of three available edges $e_{\mathbf{d}}$, $e_{\mathbf{c}}$ or $e_{\boldsymbol{\omega}}$. Pseudocodes of all essential parts of the CFP-POI algorithm follows.

The CFP-POI algorithm starts by initialization of $\mathcal{L}_{\text{set}}(v)$ and $\mathcal{L}_{\text{uns}}(v)$ for all vertices $v \in V_{\text{aux}}$ to empty sets in Lines 1–4 of Algorithm 1. The first label ℓ_s is constructed from the initial vertex s, which is also used as the vertex of the previous charging station even though it may happen that $s \notin S_{\text{aux}}$. This setting does not affect the following algorithm. The criteria function of the edge (s, s) is set to *null* since there is no such edge in E_{aux}. All the other values are set to zero because the current trip captured by the initial label has zero length, no POI stop is scheduled, and no parent label exists. Similarly, a pair $\jmath_s$ is constructed with the zero initial trip time. Both ℓ_s and $\jmath_s$ are inserted into corresponding structures $\mathcal{L}_{\text{uns}}(s)$ and $\mathcal{Q}$.

After such an initialization, the CFP-POI algorithm starts looping until the $\mathcal{Q}$ is not empty. In such a case, the empty plan $R_{\text{aux}} = ()$ is returned since the query Q_{aux} is infeasible. The following steps describe each iteration of the while loop:

- Lines 12–14 of Algorithm 1 – an extraction of a vertex-key pair $\jmath = (v, key)$ with the minimal key value from the priority queue $\mathcal{Q}$ together with extraction of the corresponding label ℓ from the priority queue $\mathcal{L}_{\text{uns}}(v)$. The label ℓ is inserted to the set of settled labels $\mathcal{L}_{\text{set}}(v)$.

- Lines 16–18 of Algorithm 1 – a detection of the destination vertex t. In the case the extracted pair $\jmath$ is located in the vertex t, the plan R_{aux} is reconstructed from the corresponding label ℓ and returned as a result.

- Lines 20–25 of Algorithm 1 – a detection and handling of the situation when the extracted label ℓ has vertex s set as the previous charging station and $s \notin S$. In that case, one new label is generated and added to corresponding $\mathcal{L}_{\text{uns}}(v)$ in case it is feasible to traverse the edge $(s, v)_{\mathbf{q}}$ meeting both consumption and POI constraints.

- Lines 26–29 of Algorithm 1 – a detection of the situation when the current vertex v of the extracted label ℓ is a charging station different from the one set in the vertex u. In that case, all feasible labels induced by charging at the previous charging station u are generated and inserted into $\mathcal{L}_{\text{uns}}(v)$. The pseudocode of the function `newLabelsInducedByCharging` is shown in Algorithm 2.

- Lines 31–36 of Algorithm 1 – a vertex-key pair $\jmath$ is not deleted when extracted from the priority queue $\mathcal{Q}$ as can be seen in Line 12. It is deleted only in the case there is no unsettled label in $\mathcal{L}_{\text{uns}}(v)$. Otherwise, the value of the key, which is minimal over all labels in $\mathcal{L}_{\text{uns}}(v)$, is set as a key of the pair $\jmath$ corresponding to the vertex v in $\mathcal{Q}$.

Algorithm 1: CFP-POI Algorithm Pseudocode

Input: $Q_{\text{aux}} = (G_{\text{aux}}, s, t, \beta_s, \beta_t, S_{\text{aux}}, L_{\text{aux}}, d_{\text{POI}}, f_{\text{POI}})$
Output: $R_{\text{aux}} = (P^{\text{aux}}_{s,t}, T_{P^{\text{aux}}_{s,t}}, Z_{P^{\text{aux}}_{s,t}})$

for $v \in V$ **do**
  $\mathcal{L}_{\text{set}}(v) \leftarrow \emptyset$
  $\mathcal{L}_{\text{uns}}(v) \leftarrow \emptyset$
end

$\ell_s \leftarrow (s, s, \beta_s, 0, null, 0, 0, 0, null)$
$\jmath_s \leftarrow (s, 0)$
$\mathcal{L}_{\text{uns}}(s)$.insert($\ell_s$)
$\mathcal{Q}$.insert($\jmath_s$)

while $\mathcal{Q}$.notEmpty() **do**
  $\jmath = (v, key) \leftarrow \mathcal{Q}$.top()
  $\ell = (v, u, \beta_u, \mathcal{T}_{\text{trip}}, \mathbf{q}, key, x_{\text{POI}}, t_{\text{POI}}, \ell_{\text{parent}}) \leftarrow \mathcal{L}_{\text{uns}}(v)$.pop()
  $\mathcal{L}_{\text{set}}(v)$.push($\ell$)

  if $v = t$ **then**
    return getPlanFromLabel(ℓ)
  end

  if $v \neq s \ \ \& \ \ u = s \ \ \& \ \ s \notin S_{\text{aux}}$ **then**
    if $\mathbf{b}_{(u,v)_{\mathbf{q}}}(\beta_s) \neq -\infty \ \ \& \ \ \mathbf{d}((u,v)_{\mathbf{q}}) + Ex_v < f_{\text{POI}}$ **then**
      $\mathcal{T}'_{\text{trip}} = \mathbf{d}((u,v)_{\mathbf{q}}) + Ex_v$
      $\ell' \leftarrow (v, v, \mathbf{b}_{(u,v)_{\mathbf{q}}}(\beta_s), \mathcal{T}'_{\text{trip}}, null, \mathcal{T}'_{\text{trip}}, 0, 0, \ell)$
      $\mathcal{L}_{\text{uns}}(v)$.insert($\ell'$)
    end
  else if $v \in S_{\text{aux}} \setminus \{u\} \ \ \& \ \ u \neq s$ **then**
    $newLabels \leftarrow$ newLabelsInducedByCharging(ℓ)
    $\mathcal{L}_{\text{uns}}(v)$.insertAll($newLabels$)
  end

  if $\mathcal{L}_{\text{uns}}(v)$.notEmpty() **then**
    $key' = \mathcal{L}_{\text{uns}}(v)$.top().getKey()
    $\mathcal{Q}$.update(v, key')
  else
    $\mathcal{Q}$.pop()
  end

  if $v = u$ **then**
    newLabelsInducedByDriving($\mathcal{Q}, \mathcal{L}_{\text{uns}}, \ell, s, t, \beta_s$)
  end
end
return ()

- Lines 38–40 of Algorithm 1 – all feasible labels induced by driving from the vertex v to some neighboring vertex w are generated together with an update of corresponding $\mathcal{L}_{\text{uns}}(w)$ and $\mathcal{Q}$. The pseudocode of the function *newLabelsInducedByDriving* is shown in Algorithm 3. Only labels satisfying the condition that the current label v is the previous charging station u, are expanded here. This condition enforces charging at each visited charging station since it is met only in the case of the initial label ℓ_s and labels produced as a result of charging. These two cases are the only cases to produce labels with equal vertices for the current vertex and the previous charging station, as can be seen in the function `newLabelsInducedByCharging` with pseudocode in Algorithm 2.

Generation of Labels at Charging Stations

Baumn et al. [6] define the concept of switching sequences for labels similar to those used in the thesis. The switching sequence is defined as a sequence of charging times $\mathcal{T}_{\text{sw}} = [\mathcal{T}_{\text{ch}_1}, \ldots, \mathcal{T}_{\text{ch}_k}]$ at the previous charging station u after which it may be beneficial to stop charging at u, drive to v and start charging there. The key is the presence of the previous charging station u, the current charging station v corresponding to the current vertex of the label ℓ and corresponding charging functions $\mathbf{cf}_u$ and $\mathbf{cf}_v$. $\mathcal{T}_{\text{ch}_1}$ is the value of the minimal charging time spent at the charging station located at the vertex u to reach the vertex v respecting all consumption constraints. The definition is formally done by introducing functions $\angle^{\ell}_{old}$ and $\angle^{\ell}_{new}$. The function $\angle^{\ell}_{old}$ maps the charging time $\mathcal{T}_{\text{ch}}$ to the slope of the charging function $\mathbf{cf}_u$ corresponding to the charging station at the vertex u after time $\mathcal{T}_{\text{ch}}$ when charging from β_u. The function $\angle^{\ell}_{new}$ maps the charging time $\mathcal{T}_{\text{ch}}$ at the charging station u to the slope of the charging function $\mathbf{cf}_v$ corresponding to the vertex v. To make functions $\angle^{\ell}_{old}$ and $\angle^{\ell}_{new}$ comparable, the function $\angle^{\ell}_{new}$ captures the slope of the $\mathbf{cf}_v$ in the moment when the EV stops charging at the charging station u after the time $\mathcal{T}_{\text{ch}}$, leaves with the corresponding amount of charged energy, drives to the charging station located in the vertex v and starts charging there.

For example, we assume a label ℓ located at the charging station at the vertex v with the previous charging station at the vertex u. Both charging stations are superchargers with a power of 175 kW. The arrival value of the SoC to the vertex u is $\beta_u = 5$ %, which implies high speed of charging as shown in Figure 3.6a and steep slope of the charging function $\mathbf{cf}_u$ shown in Figure 2.2a. However, after spending some time charging at u, the speed starts to decrease together with the slope of the charging function $\mathbf{cf}_u$. This mechanism is described by the function $\angle^{\ell}_{old}$ which captures the decreasing slope of the charging function $\mathbf{cf}_u$. On the other hand, the function $\angle^{\ell}_{new}$ captures the slope of the charging function $\mathbf{cf}_u$, at the moment when the EV stops charging at u, drives to v and starts charging there. The crucial is to determine the correct charging time $\mathcal{T}_{\text{ch}}$ to spend at the charging station u, which corresponds to the moment when it is more convenient to stop charging

there since the speed decreases and move to the charging station at the vertex v.

Baumn et al. prove, that considering piecewise linear, concave and non-decreasing charging functions **cf**, it is enough to generate one new label for each breakpoint of the charging function $\mathbf{cf}_u$ corresponding to the previous charging station at the vertex u, to preserve optimality of the algorithm. In other words, they prove that the function $\angle^{\ell}_{old}$ capturing slopes of the charging function $\mathbf{cf}_u$ after charging time $\mathcal{T}_{\text{ch}}$ can surpass the function $\angle^{\ell}_{new}$ only in values of $\mathcal{T}_{\text{ch}}$ corresponding to breakpoints of the function $\mathbf{cf}_u$. Furthermore, they show that considering piecewise linear, non-decreasing but not necessarily concave charging functions; it is enough to generate one new label for each $\mathcal{T}_{\text{ch}}$ corresponding to breakpoints of both $\angle^{\ell}_{old}$ and $\angle^{\ell}_{new}$ since only in these values one can surpass another.

In this thesis, we use results proven by Baumn et al. and we assume, that charging functions $\mathbf{cf}_v$ for all charging stations $v \in S$ are piecewise linear, concave and non-decreasing functions. This enables us to generate the switching sequence $\mathcal{T}_{\text{sw}}$ for the label ℓ, where the first charging time $\mathcal{T}_{\text{ch}_1}$ corresponds to the minimal charging time $\mathcal{T}_{\text{ch}_{\text{min}}}$ at the previous charging station at the vertex u to traverse the edge $(u, v)_{\mathbf{q}}$. All the other values $\mathcal{T}_{\text{ch}_i}$ are equal to breakpoints of the charging function $\mathbf{cf}_u$. The last charging time $\mathcal{T}_{\text{ch}_k}$ in the switching sequence $\mathcal{T}_{\text{sw}}$ is equal to the first value $\mathcal{T}_{\text{ch}_{\text{max}}}$ of the charging time when charging from β_u at the vertex u results in 100 %.

The process of generation of new labels induced by charging at the previous charging station located at the vertex u of the currently extracted label ℓ is defined by the pseudocode in Algorithm 2. After the initialization of variables used later in the function in Lines 2–5, there is a start of a loop iterating over all breakpoints of the charging function $\mathbf{cf}_u$ together with values for the minimal and maximal charging time $\mathcal{T}_{\text{ch}_{\text{min}}}$ and $\mathcal{T}_{\text{ch}_{\text{max}}}$. For each such value, the following steps are done:

- Line 8 of Algorithm 2 – a check if the current breakpoint br is inside the valid interval of charging times to induce new labels i.e., within $\mathcal{T}_{\text{ch}_{\text{min}}}$ and $\mathcal{T}_{\text{ch}_{\text{max}}}$.
- Line 11 of Algorithm 2 – a check if generating a new label ℓ' corresponding to the charging time $\mathcal{T}_{\text{ch}}$ does not violate constraints for scheduling POIs. Namely, if the interval for scheduling POIs is not surpassed or eventually it is possible to schedule the POI stop at the previous charging station u during the charging time $\mathcal{T}_{\text{ch}}$.
- Lines 15–18 of Algorithm 2 – a scheduling of the POI stop in the case there is such a POI nearby the previous charging station u and the charging time $\mathcal{T}_{\text{ch}}$ is long enough i.e., it suffices the requirement for the POI stop duration.
- Lines 20–24 of Algorithm 2 – a generation of new label ℓ' induced by the charging time $\mathcal{T}_{\text{ch}}$ at the previous charging station u and adding the label ℓ' to the vector of new labels.

Algorithm 2: Function generating labels induced by charging.

Input: $\ell = (v, u, \beta_u, \mathcal{T}_{\text{trip}}, \mathbf{q}, key, x_{\text{POI}}, t_{\text{POI}}, \ell_{\text{par}})$
Output: $newLabels$

Function `newLabelsInducedByCharging(`ℓ`):`
- $\mathcal{T}_{\text{ch}_{\min}} \leftarrow key - \mathcal{T}_{\text{trip}}$
- $\mathcal{T}_{\text{to\_init}} \leftarrow \mathbf{cf}_u^{-1}(\beta_u)$
- $\mathcal{T}_{\text{ch}_{\max}} \leftarrow \mathbf{cf}_u^{-1}(100) - \mathcal{T}_{\text{to\_init}}$
- $newLabels \leftarrow []$
- **for** $br \in \text{breakpoints}(\mathbf{cf}_u) \cup \{\mathcal{T}_{\text{ch}_{\min}} + \mathcal{T}_{\text{to\_init}}, \mathcal{T}_{\text{ch}_{\max}} + \mathcal{T}_{\text{to\_init}}\}$ **do**
  - **if** $\mathcal{T}_{\text{to\_init}} + \mathcal{T}_{\text{ch}_{\min}} \leq br \leq \mathcal{T}_{\text{to\_init}} + \mathcal{T}_{\text{ch}_{\max}}$ **then**
    - $\mathcal{T}_{\text{ch}} \leftarrow br - \mathcal{T}_{\text{to\_init}}$
    - **if** $(\mathcal{T}_{\text{trip}} + \mathcal{T}_{\text{ch}} - t_{\text{POI}} < f_{\text{POI}}) \mid (\mathcal{T}_{\text{ch}} \geq d_{\text{POI}} \ \ \& \ \ u \in L_{\text{aux}})$ **then**
      - $x'_{\text{POI}} = 0$
      - $t'_{\text{POI}} = t_{\text{POI}}$
      - **if** $\mathcal{T}_{\text{ch}} \geq d_{\text{POI}} \ \ \& \ \ u \in L_{\text{aux}}$ **then**
        - $x'_{\text{POI}} = 1$
        - $t'_{\text{POI}} = \mathcal{T}_{\text{trip}} - \mathbf{d}((u, v)_{\mathbf{q}}) + \mathcal{T}_{\text{ch}}$
      - **end**
      - $\mathcal{T}'_{\text{trip}} \leftarrow \mathcal{T}_{\text{trip}} + \mathcal{T}_{\text{ch}}$
      - $\beta'_v \leftarrow f\langle\ell\rangle(\mathcal{T}'_{\text{trip}})$
      - $key' \leftarrow \mathcal{T}'_{\text{trip}}$
      - $\ell' \leftarrow (v, v, \beta'_v, \mathcal{T}'_{\text{trip}}, key', x'_{\text{POI}}, t'_{\text{POI}}, \ell)$
      - $newLabels$.insert(ℓ')
    - **end**
  - **end**
- **end**
- **if** $u \in L_{\text{aux}} \ \ \& \ \ d_{\text{POI}} \geq \mathcal{T}_{\text{ch}_{\min}} \ \ \& \ \ \mathcal{T}_{\text{trip}} + d_{\text{POI}} - t_{\text{POI}} < f_{\text{POI}}$ **then**
  - $\mathcal{T}'_{\text{trip}} \leftarrow \mathcal{T}_{\text{trip}} + d_{\text{POI}}$
  - $\beta'_v \leftarrow f\langle\ell\rangle(\mathcal{T}'_{\text{trip}})$
  - $key' \leftarrow \mathcal{T}'_{\text{trip}}$
  - $t'_{\text{POI}} \leftarrow \mathcal{T}_{\text{trip}} - \mathbf{d}((u, v)_{\mathbf{q}}) + d_{\text{POI}}$
  - $\ell' \leftarrow (v, v, \beta'_v, \mathcal{T}'_{\text{trip}}, key', 1, t'_{\text{POI}}, \ell)$
  - $newLabels$.insert(ℓ')
- **end**
- **return** $newLabels$

- Lines 36–36 of Algorithm 2 – a generation of one more new label ℓ' in the case it is possible to schedule the POI stop at the previous charging station u for the time corresponding with the desired POI stop duration d_{POI} and neither of consumption nor POI constraints are violated.

Generation of Labels Induced by Driving Between Vertices

The second way of generation of new labels in the CFP-POI algorithm captures the driving between individual charging stations $v \in S_{\text{aux}}$, eventually between the initial vertex s and the destination vertex t.

In Lines 2–6 of Algorithm 3, there is an initialization of variables from the input label ℓ and setting of the SoC value β_{max} which stands for the maximal value of the SoC at the current vertex v. It is equal to 100 % in the case the vertex v is not a charging station, i.e., $v \notin S_{\text{aux}}$. It is important to realize that we assume only labels with the same vertices for the current vertex v and the previous charging station u, which is enforced by the condition in Line 38 in Algorithm 1. Such labels originate from the generation of new labels induced by charging at previous charging stations, which is shown in Algorithm 2. After the initialization, loop over all outgoing edges of vertex v is started with the following steps in each iteration:

- Line 9 of Algorithm 3 – a check if it is feasible to drive along the edge $(v, w)_{\mathbf{q}'}$ with the maximal possible SoC β_{max} at the vertex v.
- Lines 10–13 of Algorithm 3 – an initialization of parameters for the new label label ℓ'.
- Lines 15–15 of Algorithm 3 – in the case of traversing the edge $(v, w)_{\mathbf{q}}$ is infeasible, the value of key' corresponding to the new label ℓ' is computed as the sum of the current trip time $\mathcal{T}_{\text{trip}}$ and the time needed to spent by charging to the minimal feasible SoC β_{key} at the charging station located at the vertex v. The vertex v is necessarily a charging station at this phase of the function since the only other alternative for v which is not a charging station is the initial vertex s, but if the vertex v would be equal to the vertex s and it would be unfeasible to traverse edge $(v, w)_{\mathbf{q}}$ with the SoC value β_s, the condition in Line 9 would not pass.
- Lines 20–26 of Algorithm 3 – in the case $w = t$, e.g., the neighboring vertex w is the destination vertex t, constraints for scheduling POI in the interval f_{POI} are violated and it is possible to schedule the POI stop at the vertex v, corresponding variables x'_{POI} and t'_{POI} are updated. In the case when the minimal feasible trip time containing the POI stop is greater than the current value of the key', the key' is updated. In the case $w \neq t$, e.g., the neighboring vertex w is not the destination vertex t and the constraint for scheduling POIs in the interval f_{POI} is violated, no label is added. Labels scheduling POI stops in the case $w \neq t$ are produced in the function generating new labels induced by charging at the previous charging stations shown in Algorithm 2.

Algorithm 3: Function generating labels induced by driving.

Input: $\mathcal{Q}$, $\mathcal{L}_{\text{uns}}$, ℓ, s, t, β_s

Function `newLabelsInducedByDriving(`$\mathcal{Q}$, $\mathcal{L}_{\text{uns}}$, ℓ, s, t, β_s`):`

$(v, v, \beta_v, \mathcal{T}_{\text{trip}}, \mathbf{q}, key, x_{\text{POI}}, t_{\text{POI}}, \ell_{\text{par}}) \leftarrow \ell$

$\beta_{\max} = 100$

if $v = s$ **then**

  $\beta_{\max} = \beta_s$

end

for $(v, w)_{\mathbf{q}'} \in \text{out\_edges}(v)$ **do**

  if $\mathbf{b}_{(v,w)_{\mathbf{q}'}}(\beta_{\max}) \neq -\infty$ **then**

    $\mathcal{T}'_{\text{trip}} = \mathcal{T}_{\text{trip}} + \mathbf{d}((v, w)_{\mathbf{q}'})$

    $key' = \mathcal{T}'_{\text{trip}}$

    $x'_{\text{POI}} = 0$

    $t'_{\text{POI}} = t_{\text{POI}}$

    if $\mathbf{b}_{(v,w)_{\mathbf{q}'}}(\beta_v) = -\infty$ **then**

      $\beta_{\text{key}} \leftarrow \mathbf{c}((v, w)_{\mathbf{q}'}) + \boldsymbol{\beta}_{\min}(v) - \beta_v$

      $key' \leftarrow \mathcal{T}'_{\text{trip}} + \mathbf{cf}_v^{-1}(\beta_{\text{key}}) - \mathbf{cf}_v^{-1}(\beta_v)$

    end

    if $key' - t_{\text{POI}} > d_{\text{POI}}$ & $w = t$ & $v \in L_{\text{aux}}$ **then**

      $x'_{\text{POI}} \leftarrow 1$

      $t'_{\text{POI}} \leftarrow \mathcal{T}_{\text{trip}} + d_{\text{POI}}$

      $key' \leftarrow \max\{key', \mathcal{T}_{\text{trip}} + d_{\text{POI}} + \mathbf{d}((v, w)_{\mathbf{q}'})\}$

    else if $key' - t_{\text{POI}} > d_{\text{POI}}$ **then**

      continue

    end

    $\ell' \leftarrow (v, w, \beta_v, \mathcal{T}'_{\text{trip}}, \mathbf{q}, key', x'_{\text{POI}}, t'_{\text{POI}}, \ell)$

    $\mathcal{L}_{\text{uns}}(w).\text{insert}(\ell')$

    if $\ell' = \mathcal{L}_{\text{uns}}(w).\text{top}()$ **then**

      $\mathcal{Q}.\text{update}(w, key')$

    end

  end

end

- Lines 28–29 of Algorithm 3 – a generation of new label ℓ' and adding the label to the corresponding priority queue $\mathcal{L}_{\text{uns}}(w)$.
- Lines 31–33 of Algorithm 3 – in the case the newly generated label ℓ' has the minimal value of key' among all labels corresponding to the vertex w, the priority queue $\mathcal{Q}$ gathering vertex-key pairs is updated.

Label Dominance Checks

Up to this moment, sets $\mathcal{L}_{\text{set}}$ were used merely for adding labels corresponding to vertex-key pairs extracted from the priority queue $\mathcal{Q}$ in Algorithms 1, 2 and 3. Their main purpose is shown in label dominance checks, which are not explicitly specified in any of these algorithms. However, an essential part of the CFP-POI algorithm, same as in the case of the CFP algorithm from [6], is maintaining the invariant that priority queues $\mathcal{L}_{\text{uns}}(v)$ for all $v \in V_{\text{aux}}$ are either empty or their top label is not dominated by any settled label $\ell \in \mathcal{L}_{\text{set}}$. To avoid unnecessary dominance checks, they are done automatically in the case the top label in sets $\mathcal{L}_{\text{uns}}(v)$ corresponding with the vertex v changes as a consequence of adding or deleting labels to or from these priority queues. Such a situation may occur for example in Lines 24 and 28 in Algorithm 1 or in Line 29 in Algorithm 3.

Solution Retrieval

As soon as the first label ℓ located at the destination vertex t is extracted from the priority queue $\mathcal{Q}$, it represents the last label of the optimal solution for the EVRT-POI problem considering the auxiliary graph G_{aux}. The plan R_{aux} is retrieved by the backward iteration along pointers to labels ℓ_{par}. This leads to the gradual construction of the path $P^{\text{aux}}_{s,t}$ from vertices in corresponding labels, charging plans $T_{P^{\text{aux}}_{s,t}}$ from $\mathcal{T}_{\text{trip}}$ times of neighbouring labels and POI stop schedules $Z_{P^{\text{aux}}_{s,t}}$ from x_{POI} values determining whether the POI stop is scheduled in the previous label ℓ_{par} for the duration time d_{POI}. Such a process results in the plan:

$$R_{\text{aux}} = (P^{\text{aux}}_{s,t}, T_{P^{\text{aux}}_{s,t}}, Z_{P^{\text{aux}}_{s,t}}).$$

Simultaneously with the construction of the plan R_{aux}, it is essential to gather the sequence of criteria function markers which characterize one of the criteria functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$, which correspond to edges between two consecutive vertices u, $v \in P^{\text{aux}}_{s,t}$. Due to this sequence, it is possible to transform the path $P^{\text{aux}}_{s,t}$ of vertices $u \in V_{\text{aux}}$ to the path $P_{s,t}$ of vertices $v \in V$ in the original route graph G. It is done similarly as in the case of the ILP model by running the shortest path queries with the corresponding criteria $\mathbf{q}$ between all neighbouring vertices u, $v \in P^{\text{aux}}_{s,t}$. The charging plan $T_{P_{s,t}}$ and the POI sop schedule $Z_{P_{s,t}}$ are constructed by matching of vertices $u \in P^{\text{aux}}_{s,t}$ to corresponding vertices $v \in P_{s,t}$. This results in the final plan R, which is the solution of the original problem EVRT-POI defined in Section 2.2:

$$R = (P_{s,t}, T_{P_{s,t}}, Z_{P_{s,t}}).$$

Heuristic Functions to Speedup the Search

The rationale behind the introduction of heuristics in the CFP-POI algorithm follows the concept of the A* algorithm. The key is the introduction of a potential function $\mathbf{h}$ which provides an estimate of the time needed to reach the destination location. Contrarily to the optimization of a single criterion in the case of the EVRT-POI problem, it is essential to incorporate the current SoC, which implies the estimate of the time needed to be spent by charging. In the case of the CFP-POI, values of the heuristic function $\mathbf{h}$ for each label ℓ are added to the value of *key*, which changes their priority in the priority queue $\mathcal{Q}$ and increases the probability of being extracted to these labels which have shorter estimated total time of the complete plan R_{aux}.

Single-Criterion Search Heuristic. The initial heuristic used in the thesis is the one proposed by Baum et al. in [6]. It is based on the criteria function $\boldsymbol{\omega}$ defined in Subsection 3.2.1. The criteria function $\boldsymbol{\omega}$ assigns each edge $e_{\mathbf{q}} \in V_{\text{aux}}$ (potentially $e \in V$) the value of the time $\mathbf{d}(e_{\mathbf{q}})$ needed to traverse the edge plus the time needed to recharge the amount of energy consumed by the traversal using the charging station with highest possible constant speed of charging. Furthermore, it uses shortest paths between the current vertex v and the destination vertex t optimizing criteria functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$. Due to the construction of the auxiliary graph G_{aux} introduced in Section 3.2, we have all these values available by the time of the CFP-POI algorithm starts. The heuristic potential function $\pi_{\boldsymbol{\omega}} : V_{\text{aux}} \times [0, 100] \to \mathbb{R}_{\geq 0}$ is defined as follows:

$$\pi_{\boldsymbol{\omega}}(v, \beta_{\text{key}}) := \begin{cases} \mathbf{d}((v,t)_{\mathbf{d}}) & \text{if } \beta_{\text{key}} \geq \mathbf{c}((v,t)_{\mathbf{c}}) + \boldsymbol{\beta}_{\min}(t) \\ \mathbf{d}((v,t)_{\boldsymbol{\omega}}) - \frac{\beta_{\text{key}}}{\mathbf{cf}_{\max}} + \frac{\boldsymbol{\beta}_{\min}(t)}{\mathbf{cf}_{\max}} & \text{otherwise.} \end{cases}$$

Here, the value of β_{key} is the minimal SoC at the current vertex v of the label ℓ, which corresponds to the minimal charging time at the previous charging station u and traversing the edge $(u, v)_{\mathbf{q}}$ without violation of consumption constraints. The value of the potential function $\pi_{\boldsymbol{\omega}}(v, \beta_{\text{key}})$ is equal to the shortest path optimizing time criteria $\mathbf{d}$ from the current vertex v to the destination vertex t in the case the minimal SoC β_{key} is greater than the most energy-efficient shortest path between these two vertices. Otherwise, the value is retrieved as the time needed to traverse the shortest path optimizing the criteria function $\boldsymbol{\omega}$ minus the time equal to the charging time from zero percent to the minimal SoC β_{key}. Additionally to the potential function $\pi_{\boldsymbol{\omega}}$ introduced in [6], we incorporate constraints capturing the the minimal SoC $\boldsymbol{\beta}_{\min}(t)$ at the destination vertex t. Therefore, we add the estimated time of charging this amount of energy on the fastest possible charging station into the second case in the definition. The meaning of the subtraction of $\frac{\beta_{\text{key}}}{\mathbf{cf}_{\max}}$ is that as the value of $\mathbf{d}((v,t)_{\boldsymbol{\omega}})$ contains estimated charging time on all edges on the path between v and t, there is no need to charge the amount of energy the EV already has in β_{key}.

Plan Precomputation-Based Heuristic. Besides the heuristic based on the potential function π_{ω}, we introduce the heuristic approach estimating the total time of the plan including the driving and charging time from the current vertex v to the destination vertex t. The estimation is done by times of precomputed plans between specific locations. The process of the computation of values of our plan Precomputation-Based heuristic consists of the following steps:

1. Determining the set $\mathcal{K} \subseteq V$ of vertices $v \in V$ of the size $k \in \mathbb{N}$. Vertices in $\mathcal{K} \subseteq V$ represent the spots in between which the plans will be further precomputed. The criteria for selecting vertices $v \in V$ should reflect the probability of these vertices being used for searching the shortest plans later on during the runtime of the CFP-POI algorithm. Since the CFP-POI algorithm operates on the auxiliary graph G_{aux} consisting predominantly of charging station vertices, and it searches for heuristic values in them, we propose to apply the k-means algorithm to cluster the set of charging stations L_{aux} into k clusters. Set $\mathcal{K}$ corresponds to the set of k centroids $ce_i \in V$, $i \in \{1, ..., k\}$. Similarly, as in the case of using the k-means algorithm for selecting charging stations in corridors in Subsection 3.2.2, centroids ce_i do not necessarily correspond with any vertex $v \in V$. Hence we identify them with the closest vertex concerning the haversine distance.

2. Precomputing of times of plans between individual vertices $v \in \mathcal{K}$ using the CFP-POI algorithm with π_{ω} heuristic function. The total time of the plan between two vertices u, $v \in \mathcal{K}$ is affected by the initial SoC β_s and the minimal destination SoC β_t. Therefore, we propose precomputing one plan for each combination of values of the initial SoC bins $\beta_{\text{in}} = [10, 20, \ldots, 100]$ and the destination SoC bins $\beta_{\text{dest}} = [0, 10, \ldots, 80]$ between each pair of vertices u, $v \in \mathcal{K}$. We do not consider 0 % SoC for bins in β_{in} and values of 90 and 100 % for bins in β_{dest} since this would lead to frequent occurrence of infeasible plans.

3. Introducing the potential function:

$$\mu : V_{\text{aux}} \times V_{\text{aux}} \times [0, 100] \times [0, 100] \to \mathbb{R}_{\geq 0}.$$

   This function maps the current vertex $v \in V_{\text{aux}}$, the destination vertex $t \in V_{\text{aux}}$, the SoC value β_v at the vertex v and the minimal SoC value β_t at the vertex t to the estimated time of the plan from the vertex v to the vertex t including driving and charging time. The value of the potential function μ is retrieved as follows:

   a. Search for the vertices $u \in \mathcal{K}$ with the haversine distance from the vertex v in the range $\delta \in \mathbb{R}_{\geq 0}$. At the same time searching for the vertices $w \in \mathcal{K}$ with the haversine distance from the vertex t in the same range δ. Such vertices are collected in the set $\mathcal{O}$ for the vertex v and in the set $\mathcal{P}$ for the vertex t.

b. Choose the vertex $a \in \mathcal{O}$ with the minimal haversine distance to the destination vertex t and the vertex $b \in \mathcal{P}$ with the minimal haversine distance to the vertex v. This step heuristically estimates the direction of the path from v to t and analogically from t to v which prevents choosing vertices from sets $\mathcal{O}$ and $\mathcal{P}$ located in the opposite directions. Such a wrong choice could lead to the overestimation of the driving time in the heuristic value.

c. Assuming the following label ℓ in the CFP-POI algorithm:

$$\ell = (v, u, \beta_u, \mathcal{T}_{\text{trip}}, \mathbf{q}, key, x_{\text{POI}}, t_{\text{POI}}, \ell_{\text{par}}).$$

We use the precomputed value of the plan between chosen vertices $a \in \mathcal{O}$ and $b \in \mathcal{P}$ corresponding to the SoC bins $\lceil \beta_{\text{key}} \rceil$ and $\lfloor \beta_{\text{t}} \rfloor$ as heuristic values which are added to the value of the key and. By $\lceil \beta_{\text{key}} \rceil$ and $\lfloor \beta_{\text{t}} \rfloor$ we mean rounding up and down to corresponding SoC bins in β_{in} and β_{dest} which prevents from the overestimation of the needed charging time in the heuristic estimate.

Chapter 4

Experiments

In the following sections, we provide several experiments to evaluate individual parts of the proposed solution. We point out the advantages and disadvantages of using heuristic approaches throughout solving EVRT-POI based on gathered results. The first section describes the implementation details of proposed techniques and datasets used for the evaluation. Furthermore, we compare quantitative results from the ILP model with the CFP-POI algorithm in the region of the Czech Republic with differing settings for the minimal initial and destination SoC. Then, we perform four different experiments using queries between regions in six different European countries:

- **Germany** – **Italy** – an evaluation of the dependency of the initial and destination SoC on query times.
- **Poland** – **Switzerland** – an evaluation of the influence of the parameter $o \in \mathbb{N}$, which stands for the maximal number of chargers chosen per corridor $\mathcal{C}^{\epsilon}_{\mathbf{q}_{s,t}}$ on the query time and the quality of resulting plans.
- **Denmark** – **Croatia** – an evaluation of the influence of the presence of consumption function optimizing edges $e_{\mathbf{c}} \in E_{\text{aux}}$ and $\boldsymbol{\omega}$ criteria function optimizing edges $e_{\boldsymbol{\omega}} \in E_{\text{aux}}$ in the auxiliary graph G_{aux} on the query time and the quality of resulting plans.
- **Germany** – **Italy** – an evaluation of the dependency of the integrated POI planning on the query time with fixed initial and destination SoC.

We also evaluate and discuss the performance of the Precomputation-Based heuristic on several concrete plans. At the end of this chapter, we provide a qualitative comparison of plans produced by our CFP-POI algorithm using the auxiliary graph with plans given by ABRP. We focus on the plan's query time and overall quality from the driver's perspective. Finally, we compare the results of our algorithms with results of the CFP algorithm presented in [6]. For all of the experiments, we use parameters of Skoda Enyaq iV to model the consumption as described in Section 3.3.

4.1 Implementation

For the evaluation, we implemented all components introduced in Chapter 3 in two interconnected layers. The first layer is represented by the C++ project consisting of the following components:

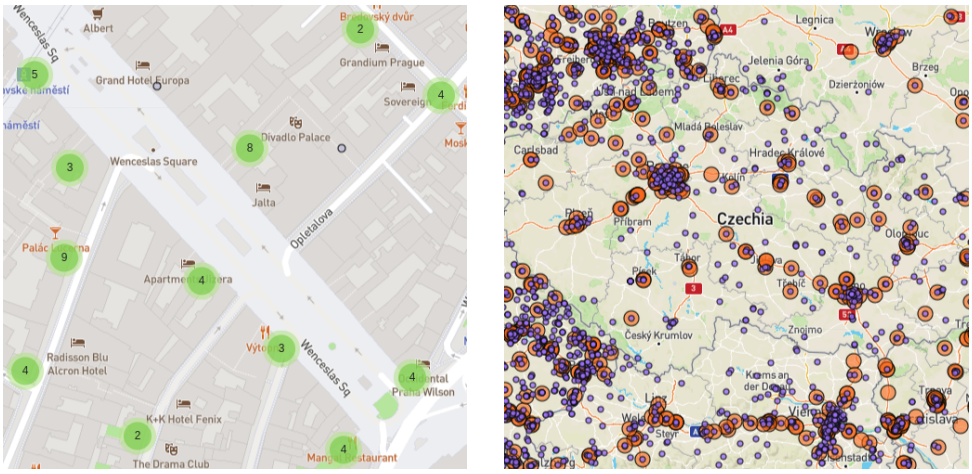
- Routines building the route graph G from an Open Street Map dataset described in Subsection 4.2.1.
- Enrichment of Route graph by elevation data mined from Copernicus satellite dataset described in 4.2.3.
- Initialization and building contraction hierarchies for the shortest path queries. RoutingKit library implemented by Dibbelt et al. [12] is used as a black box solver for the shortest path queries during the preprocessing phase and construction of the auxiliary graph.
- Implementation of the preprocessing phase, which contains precomputation of distance matrices between charging stations and precomputing charging functions $\mathbf{cf}_v$ for all charging stations $v \in S$.
- Implementation of the CFP-POI algorithm.
- TCP socket server handling requests for the shortest path and for solution plans to EVRT-POI problem. The implementation of the TCP server is inspired by the code published on the web [3]. Requests are in JSON format.

The second layer is formed by the collection of Python scripts providing the following functionality:

- Implementation of all experiments introduced later in Chapter 4.
- Implementation of the application programming interface (API) for shortest path and CFP-POI queries in JSON format.
- Implementation of precomputation of plans for Precomputation-Based heuristic.
- Implementation of the auxiliary graph construction.
- Implementation of the ILP model using primitives provided by the Gurobi solver [24].

4.2 Used Datasets

The implemented solution processes several datasets essential for real-world modeling conditions. We used open-source data to construct the route graph, to mine the set of charging stations S and to enrich the route graph by elevation data.



(a) : Restaurants in the city centre of Prague.

(b) : Chargers extracted from Open Charge Map.

Figure 4.1: Restaurants and charging stations dataset visualizations.

4.2.1 Open Street Maps

To build a road graph G used for construction of the auxiliary graph G_{aux} and for solving the shortest path problem, we use an open-source map dataset Open Street Maps (OSM) [41]. In addition to graph building, we use OSM also for mining of POIs, which serve as a data source for the enrichment of the charging station dataset described in the following Subsection 4.2.2. For POIs mining from OSM, we use the Osmosis command-line application for processing OSM data. An example of mined restaurants in the city center of Prague is shown in Figure 4.1a.

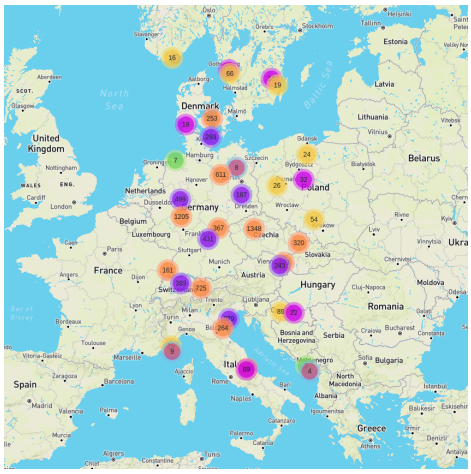
4.2.2 Charging Station Datasets

To gather information on charging stations, we use the dataset capturing the location (latitude, longitude), type (AC, DC), charging speed measured in kW, and often some additional information regarding the prize or location description. The dataset is available online through Open Charge Map API [40]. An example of charging stations in the Czech Republic retrieved from Open Charge Map API is shown in Figure 4.1b.

4.2.3 Elevation Profile

For modeling an elevation profile, we use data from a project of the European Space Agency, which publicly provides records from the Copernicus satellite in Geotiff format. These records enable enrichment of the route graph G namely vertices $v \in V$ by values of meters above sea level (m.a.s.l.). Data from Copernicus feature 25m resolution with a vertical accuracy ± 7 m. An example of an elevation map that originates from the enrichment process is in Figure 4.2a.

(a) : Elevation map extracted from Copernicus satellite data.



(b) : Area used for the evaluation of the CFP-POI algorithm performance.

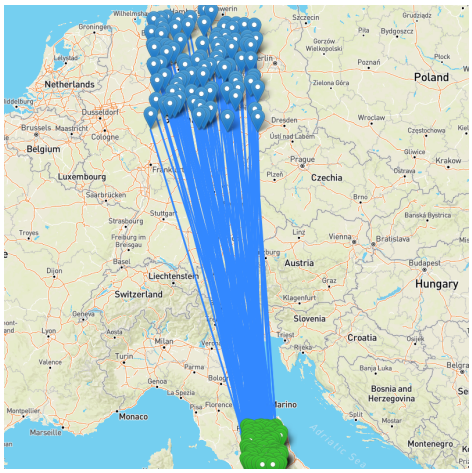
Figure 4.2: Visualization of used datasets.

4.3 Quantitative Experiments

All experiments presented in this section follow the common structure and they were conducted on a single core of Intel(R) Xeon(R) Silver 4110 CPU running at 2.10GHz with capacity of RAM 196.75 GB. Especially experiments operating on regions of European countries require approximately 25 GB of RAM because of larger route graph and amount of precomputed parameters.

We define two bounding boxes in each experiment capturing two regions on the map. The bounding box in our interpretation is fully characterized by latitude and longitude of two locations, namely coordinates determining the top left corner of the bounding box and coordinates determining the bottom right corner of the bounding box. Here we relax the curvature of the Earth for simplicity and assume that the bounding box defined by two coordinates describes the rectangular region. Having two regions delimited by bounding boxes, we randomly sample a vector of 100 locations within each region. An example of such a selection of locations within the bounding box is shown in Figure 4.3a. Corresponding locations in two vectors imply 100 pairs of initial vertices s, and destination vertices t, which determine 100 queries to run in each experiment. An example of sampled locations connected by straight lines defining the connection between the corresponding initial vertex s and the destination vertex t is shown in Figure 4.3b. Furthermore, we assume the charging curve **cc** corresponding to the one in Figure 3.6a which corresponds to Skoda Enyaq iV 80 and is transformable to the piecewise linear concave charging functions $\mathbf{cf}_v$, $\forall v \in S$ shown in Figure 2.2a by applying methods described in Section 3.4. In all experiments except for the evaluation of the Precomputation-Based heuristic in Subsection 4.3.2, we use the CFP-POI algorithm with the Single-Criterion Search heuristic.

(a) : Example of selected locations within a bounding box.



(b) : Example of testing set of paths between northern Germany and central Italy.

Figure 4.3: Example of testing set of locations and paths between northern Germany and central Italy.

4.3.1 Comparison of CFP-POI Algorithm and ILP Model

In the first set of experiments, we compare the performance and the overall quality of plans generated by the baseline ILP model and the CFP-POI algorithm. We consider the total query time, which consists of the time for the auxiliary graph construction and the computation time of the optimization algorithm, as the principal measure of performance. Regarding the plan's quality, we consider the overall time of produced plans, time spent by charging, number of charging stops, and average energy consumption. In following experiments we use parameter settings shown in Table 4.1.

$ls \in \mathbb{R}_{\geq 0}$ – the linearity interval of charging functions $\mathbf{cf}_v$	1 min
$o \in \mathbb{R}_{\geq 0}$ – the number of charging stations per corridor	150
$M \in \mathbb{R}_{\geq 0}$ – the capacity of EV battery	77 kWh
$Ex_v \in \mathbb{R}_{\geq 0}, \forall v \in S$ – the manipulation and waiting time	5 min

Table 4.1: Parameters setting for comparison of CFP-POI and ILP Model.

Due to the performance limitations of the ILP model, we perform the first set of experiments on regions within the Czech Republic. The first two bounding boxes capture the area surrounding the city of Prague and the city of Brno, which implies driving distances of around 200 km on average. Considering the consumption model of Skoda Enyaq iV, sufficient initial SoC β_s above 80 % and low values of the minimal destination SoC β_s below 10 %, it is possible to traverse such a distance without the need of charging stop. The second pair of locations is represented by regions of western Bohemia and eastern Moravia with an approximate distance of 550 km, which enforces the planner

to schedule a charging stop.

In the case of regions surrounding Prague and Brno, we perform a single experiment with the initial SoC $\beta_s = 90$ % and the destination SoC $\beta_t = 10$ % to compare the performance and plan quality of the CFP-POI algorithm and the ILP model with no need of charging stops. In the case of the western and eastern regions of the Czech Republic, we run each of 100 queries three times with different settings for the initial and the destination SoC β_s and β_t. The combinations of SoC values are written in Table 4.2.

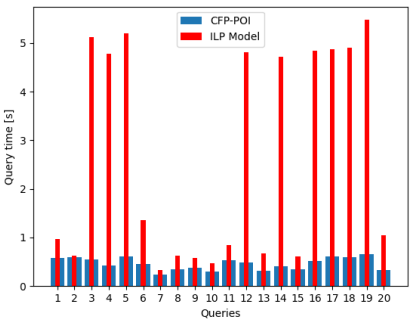
β_s [%]	β_t [%]
90	10
100	80
10	80

Table 4.2: Combinations of the initial and the destination SoC.

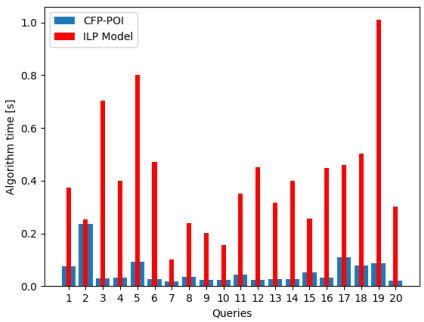
Prague and Brno Regions

Results for queries between locations surrounding Prague and Brno with no need for charging stops showed that the CFP-POI algorithm considerably outperforms the ILP model in the criteria of runtime. In Figure 4.4a on the x axis, there are blue bars for the CFP-POI algorithm and red bars for the ILP model for the first 20 queries. There is the query time on the y axis measured in minutes for each pair of bars. We choose the first 20 queries for the sake of the chart clarity, results of complete set of 100 queries are in Table A.1. Figure 4.4a shows that the proportion between queries, where the query time difference is only slightly better for the CFP-POI and queries, where the dominance of the CFP-POI is more pronounced, is approximately 50 %. The mean difference of query times between the ILP model and the CFP-POI algorithm on all 100 queries is 1.42 s in favor of the CFP-POI. The difference in the overall performance of the ILP model and the CFP-POI is even more significant when pointing out the fact that the ILP model works merely with single edges, optimizing the driving time criteria $\mathbf{d}$ in the auxiliary graph. On the other hand, queries solved by the CFP-POI algorithm work with all three edges corresponding to criteria functions $\mathbf{d}$, $\mathbf{c}$ and $\boldsymbol{\omega}$. This affects mainly the part of the query time taken by the auxiliary graph construction, which takes the majority of the total query time in the case of short query distances with no need to charge. This claim is supported by Figure 4.4b, which shows the comparison of runtimes of the ILP model and the CFP-POI without the auxiliary graph construction.

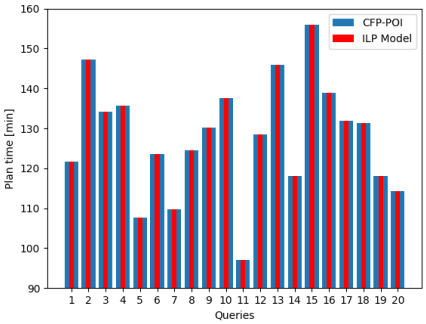
As shown in Figure 4.4c, which captures the comparison of the time of the first 20 solution plans retrieved from the two optimization methods, there are no differences in the quality of plans considering the overall plan time, which consists exclusively from the driving time in this case. The same situation remains for all the other plan quality parameters, such as the distance of individual paths or average energy consumption, which shows that



(a) : Query times.



(b) : CFP-POI runtimes.



(c) : Times of plans.

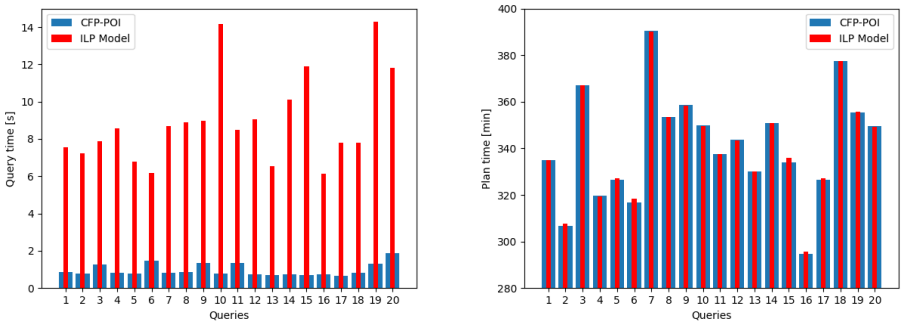
Figure 4.4: Performance and plan quality statistics for the CFP-POI and the ILP model on Prague and Brno regions with $\beta_s = 90$ and $\beta_t = 10$.

additional edges optimizing the consumption criteria $\mathbf{c}$ and the criteria $\boldsymbol{\omega}$ do not bring any improvement here.

Western and Eastern Part of the Czech Republic

Contrarily to the previous experiment in Subsection 4.3.1, here we consider queries on more considerable distances, which implies the need to schedule charging stops on the path. As shown by results presented below, more extensive distances between the initial location s and the destination location t also bring the higher potential of being used for multiple edges, optimizing multiple criteria functions in the case of the CFP-POI algorithm.

Figure 4.5a shows that differences in the performance between the ILP model and the CFP-POI algorithm considerably increase with the scheduling of charging stops. The mean difference between the query time for all 100 queries rises to the value of 8.11 seconds compared to the 1.42 in the case of the Prague and Brno regions. The mean query time of the CFP-POI algorithm on this dataset is below one second on the value 0.98 s, whereas the mean query time of the ILP model solution is 9.09 s, which means it is approximately ten times slower.



(a) : Query times.

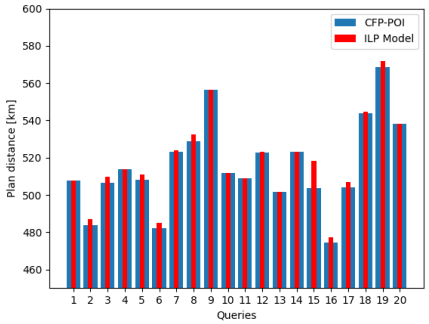
(b) : Plan times.

Figure 4.5: Performance and plan quality statistics for the CFP-POI and the ILP model on regions of western and eastern part of the Czech Republic with $\beta_s = 90$ % and $\beta_t = 10$ %.

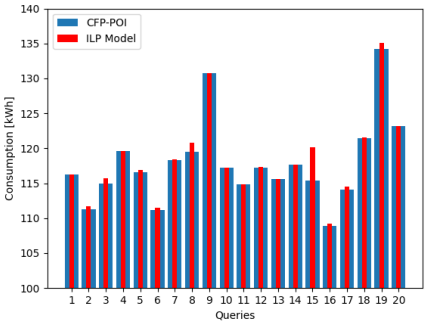
Figure 4.5b, which shows the comparison of plan times, indicates that there are only slight differences in units of minutes in favor of the CFP-POI algorithm. However, even though the differences are relatively moderate, the phenomenon behind them is pretty complex. The following four figures explain:

- Figure 4.6a – shows the first 20 queries with blue bars for the CFP-POI algorithm and red bars for the ILP model on the x axis and the total distance of each trip measured in kilometers on the y axis.
- Figure 4.6b – shows the total energy consumption measured in kWh of each trip for the same set of the first 20 queries.
- Figure 4.6c – shows the time spend by charging measured for the same set of the first 20 queries.
- Figure 4.6d – shows the number of charging stops in the same set of the first 20 queries.

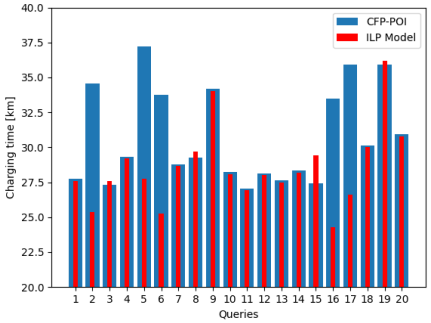
Figure 4.6a shows differences between the CFP-POI algorithm and the ILP model in distances of paths corresponding to individual queries. It applies that the ILP model's plan's distances are greater or equal to distances of paths returned by the CFP-POI algorithm in the set of 20 studied queries. This indicates that more consumption-friendly edges optimizing one of the criteria functions $\mathbf{c}$ and $\boldsymbol{\omega}$ are used in the CFP-POI solution, and they are shorter than the fastest edges produced by the optimization of the driving time. This was confirmed even by the analysis of returned plans. By the comparison of corresponding queries in Figure 4.6a and Figure 4.6b, it is possible to see that paths with the higher overall distance have also lower consumption. This is caused by the lower speeds along these longer paths, which leads to lower energy consumption. For example for queries 3, 8 or 15 the lower consumption



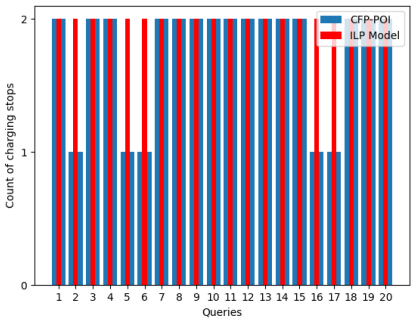
(a) : Plan length.



(b) : Energy consumption.



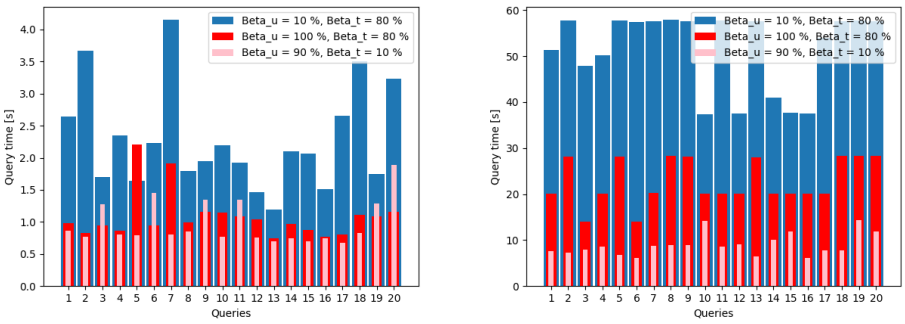
(c) : Charging time.



(d) : Count of charging stops.

Figure 4.6: Plan quality statistics for the CFP-POI and the ILP model on regions of western and eastern part of the Czech Republic with $\beta_s = 90$ % and $\beta_t = 10$ %.

of the CFP-POI path indicated in Figure 4.6b leads to shorter charging time in Figure 4.6c, which immediately implies the small difference in the the total time of the plan in Figure 4.5b. The time saved during the charging process compensates for the time spent on the longer and more consumption-friendly sub-path. On the other hand, in the case of queries 2, 5, 6, 16, and 17, the charging time in the CFP-POI plan is almost ten minutes longer than the one produced by the ILP model. Figure 4.6d shows that the same set of queries has only one charging stop scheduled in the case of the CFP-POI, whereas the ILP model schedules two charging stops. Used charging curve from Figure 3.6a suggests that using two shorter charging stops may lead to a shorter overall charging time, as the speed of charging in lower values of the SoC is higher, and this is also the cause of the low charging time in the case of the ILP model. However, two charging stops also mean two extra manipulation and waiting penalties Ex_v, which is set to 5 minutes. This parameter causes that in the global measure, the CFP-POI algorithm stays charging longer time on one charging station, even though with a slower speed because the second extra penalty is not needed and the total time of the plan is shorter.



(a) : CFP-POI query times. **(b) :** ILP model query times.

Figure 4.7: Comparison of query times of the CFP-POI algorithm and the ILP model with different settings for the initial SoC β_s and the destination SoC β_t.

The phenomena described in the previous paragraph show the importance of modeled features from the proper charging curve model, through the existence of multiple edges in the auxiliary graph up to the modeling of manipulation and waiting times in charging stations.

Furthermore, we study the dependency of the initial SoC β_s and the destination SoC β_t on the query time separately for the CFP-POI algorithm and the ILP model. Figures 4.7a and 4.7b show that there is a clear dependency between an amount of energy needed to charge and the query time. In both cases, queries with the initial SoC $\beta_s = 10$ % and the destination SOC $\beta_t = 80$ %, which implies the highest need to charge leading to 5 charging stops on average, take the longest time. The ILP model provides a considerable difference even between the second pair of settings, 100-80 and 90-10. This is not the case with the CFP-POI algorithm, where the difference is not so significant, but the statement still holds when considering the average case. It is also essential to point out that to keep the computation time of experiments reasonable, we use the upper bound for the query time of the ILP model, which is applied to queries with a higher need for charging stops. The value of the upper bound is proportional to the suboptimality gap and may rise to approximately 40 seconds in the case of the optimality gap higher than 10 %. This leads to the conclusion that the query time of the ILP model would be even higher when letting the solver find the optimal solution.

4.3.2 CFP-POI Performance Evaluation

The following subsections are dedicated to the detailed evaluation and analysis of the CFP-POI algorithm. Unless specified otherwise, the setting of parameters is the same as in the case of experiments in previous Subsections 4.3.1 and 4.3.1 and it is stated in Table 4.2. For the sake of saving precomputation time during the construction of the distance matrix described in Subsection 3.2.3, we filtered the slowest chargers with the power lower or equal to 22 kWh from the set of all chargers parsed from the Open Charge

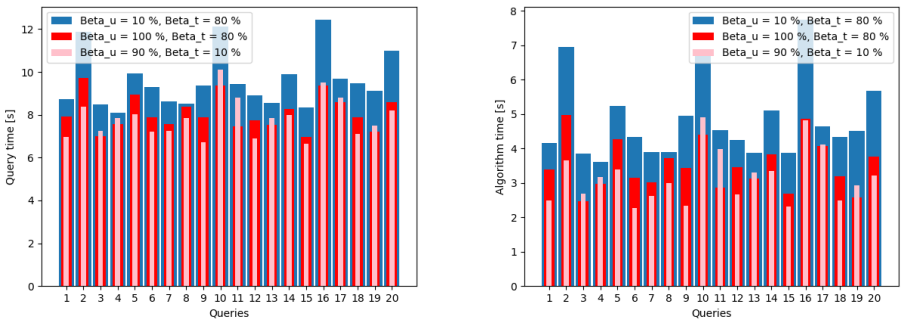
Map API. Preliminary test showed, that this filtering has no effect on results of queries with distances as high as used in our experiments, since we optimize the total time and incorporation of slow chargers into plans leads to considerable prolongations. An amount of chargers on the area marked in Figure 4.2b after filtering is 7005 which is comparable to 13810 chargers considered for the area of entire Europe in [6].

Influence of Different Initial SoC β_s and Destination SoC β_t on Query Time

Similarly, as the last experiment presented in Subsection 4.3.1, the first experiment on data of the continental-scale examines the influence of the initial value of the SoC β_s and the minimal value of the destination SoC β_t to the query time. The set of 100 queries corresponds to paths sketched in Figure 4.3b between randomly chosen starting locations in the region of northern Germany and destination locations placed in central Italy. Figure 4.8a illustrates that the dependency between individual configurations of the initial and the destination SoC does not change with more considerable distances of paths. Similarly, as in the case of Figure 4.7a, SoC configurations with higher energy demand and implicitly the higher number of charging stops require a longer query time. Values for mean query time consisting of the time spend by the auxiliary graph construction and the runtime of the CFP-POI algorithm itself are in Table 4.3. Table 4.3 also contains the part of the query time taken by the runtime of the CFP-POI algorithm without the auxiliary graph construction. From the query time and the runtime of the CFP-POI algorithm follows that the time spent by the auxiliary construction is similar for all three settings of SoC values, and it takes approximately 5 seconds. The value of mean path distance in Table 4.3 suggests that the different sets of the SoC configuration influence the length of the path as well. This may be caused by using different charging stations of edges optimizing the different criteria functions in the auxiliary graph.

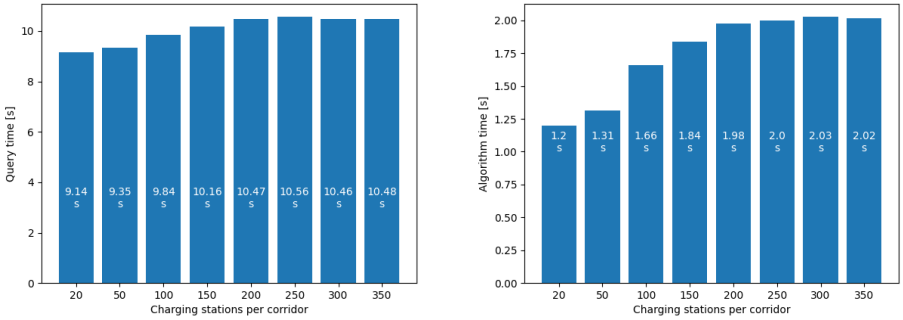
	$\beta_s = 90, \beta_t = 10$	$\beta_s = 10, \beta_t = 80$	$\beta_s = 100, \beta_t = 80$
Query time	7.70 s (±0.79)	10.15 s (±0.66)	8.21 s (±0.80)
CFP-POI runtime	3.20 s (±0.88)	5.01 s (±1.19)	3.85 s (±0.95)
Plan time	16.2 h	18.17 h	17.27 h
Distance	1454 km	1468 km	1458 km
Charging stops	6.5	9	7.5
Charging time	154 min	238 min	201 min

Table 4.3: Mean values of parameters of plans retrieved by using the CFP-POI algorithm on 100 queries between locations in northern Germany and central Italy.



(a) : CFP-POI query times. **(b) :** CFP-POI algorithm runtimes.

Figure 4.8: Comparison of query times and algorithm runtime of the CFP-POI algorithm with different settings for the initial SoC β_s and the destination SoC β_t.



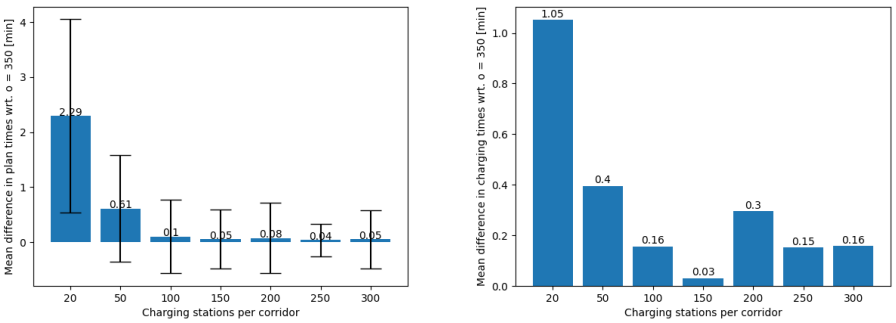
(a) : Mean CFP-POI query times. **(b) :** Mean CFP-POI algorithm runtimes.

Figure 4.9: Comparison of query times and algorithm runtime of the CFP-POI algorithm with different values of parameter $o \in \mathbb{R}$ for the maximal number of charging stations selected per corridor.

Influence of Auxiliary Graph Size on Plan Quality

The essential purpose here is to analyze the influence of the parameter $o \in \mathbb{N}$ determining the maximal number of selected charging stations per one corridor on the query time and the overall quality of plans. Experiments are done on randomly chosen locations between regions of central Poland and western Switzerland with the mean distance of the trip equal to 1105 km.

Figures 4.9a and 4.9b show that with the rising value of the parameter o, the value of the query time rises as well. The rise here is caused dominantly by the rise of runtimes of the CFP-POI algorithm, as it operates on the larger auxiliary graph G_{aux}. The most significant increase is apparent from the value of $o = 20$ up to $o = 200$, where the difference in mean values of the CFP-POI algorithm runtime for 100 queries is equal to 0.78 seconds. However, from the value of $o = 200$, the runtime stabilizes and remains



(a) : Mean differences in plan times.

(b) : Mean differences in charging times.

Figure 4.10: Comparison of mean differences between plan times and charging times for different values of parameter $o \in \mathbb{R}$ for the maximal number of charging stations selected per corridor with respect to the value of $o = 350$.

around 2 seconds. We attribute this phenomenon to the fact that at the point of 200 charging stations selected per single corridor, charging stations with lower speeds and mainly with inconvenient locations regarding the path determining the corridor are chosen. Since the CFP-POI ranks individual labels concerning the estimated time to the destination using the Single-Criterion Search heuristic approach, these labels have a lower probability of being extracted from the priority queue. Hence, they do not slow down the algorithm.

Even though the different settings of o cause the difference in query times, the crucial is its influence on the plan's overall quality. Similarly, as in the case of previous experiments, we consider the overall time of returned plan as the critical significance of the plan quality as the goal is to minimize it. We assume that the higher value of o is, the higher is the quality of returned plans because there is a lower probability of omitting some charging station $v \in S$, which would be the part of the optimal solution in the case of using all charging stations in S. Figure 4.10a shows on the x axis individual values of the parameter o and on the y axis, there are values of mean differences in times of plans for corresponding queries compared to plan times of queries received for the value of $o = 350$. The value of the mean difference between plan times sharply decreases with the first two values of $o = 20$ and $o = 50$. With all following values, the mean difference varies in negligible units of seconds, which are far below the accuracy of our computations as we consider one minute as the minimal unit for a driving time along edges and for charging at charging stations. Such a result suggests that the value of $o = 100$ is sufficient for preserving the quality of plans returned from the CFP-POI algorithm. However, the mean difference of 2.29 minutes in the case of $o = 20$ seems to be a reasonable trade-off between the accuracy and query time reduction when considering the mean time of plans being equal to 12.6 hours.

Furthermore, Figure 4.10b shows the mean differences between charging times of plans with different settings of the parameter o with the maximal value of $o = 350$. It indicates the similar decreasing trend as in the case of

mean differences of plan paths. A possible interpretation of this phenomenon is that the decreasing number of selected charging stations per corridor increases the probability of non-selecting favorable charging stations (fast-charging stations with convenient locations with respect to the corridor path). This leads to the need to spend a longer time charging at one charging station, and the consequences are a worse ratio between the charging time and an amount of charged energy as suggested by the charging curve in Figure 3.6a. It is true that the value of the mean charging time for $o = 200$ in Figure 4.10b may evoke a violation of previously discussed trends. However, we consider the value of 0.3 min still bellow the threshold of accuracy of our computations. Furthermore, detailed analysis of results showed that this artifact is caused by a few outliers in charging times, which do not affect the overall time of produced plans as can be seen in Figure 4.10a.

Influence of Consumption and Single Potential Edges in Auxiliary Graph on Plan Quality

To determine the influence of the presence of edges optimizing criteria functions $\mathbf{c}$ and $\boldsymbol{\omega}$ in the auxiliary graph, we used the set of 100 queries between the region of central Denmark and northern Croatia with the mean path distance of approximately 1760 km and the mean plan time around 19.4 hours. The mean distance between the initial and destination locations in queries used for this experiment is the longest among all the other queries. The value of the initial SoC is set to $\beta_s = 90$ % and the minimal destination SoC $\beta_t = 10$ %.

	$\mathbf{d}$, $\mathbf{c}$, $\boldsymbol{\omega}$	$\mathbf{d}$, $\boldsymbol{\omega}$	$\mathbf{d}$
Query time	8.22 s (±0.52)	6.20 s (±0.33)	5.04 s (±0.26)
CFP-POI runtime	3.15 s (±0.44)	2.28 s (±0.28)	2.16 s (±0.23)
Plan time	19.38 h	19.38 h	19.43 h
Distance	1753 km	1753 km	1771 km
Charging stops	8.8	8.8	8.8
Charging time	188.60 min	188.60 min	192.22 min

Table 4.4: Mean values of parameters of plans retrieved by using the CFP-POI algorithm on 100 queries between locations in central Denmark and northern Croatia with differing presence of edges in the auxiliary graph G_{aux}.

Duplicate values for plan time, distance, an amount of charging stops, and charging time in the first two columns of Table 4.4 indicate that adding edges optimizing the consumption criteria function $\mathbf{c}$ to the auxiliary graph G_{aux} does not have any effect on the final plan. This is caused by a high trade-off between consumption and time efficiency of planned paths, which makes energy-efficient edges rather inconvenient for time optimization problems. Omitting edges optimizing the consumption criteria $\mathbf{c}$ also leads to the decrease of the mean query time by almost 2 seconds. On the other hand, omitting edges optimizing the criteria function $\boldsymbol{\omega}$, which represents a trade-off between the driving time and the energy consumption along edges,

leads to a prolongation of the total time of plans by 3 minutes on average. The maximal prolongation of the plan is 13.21 minutes. Considering these results and the construction of the criteria function $\boldsymbol{\omega}$ in Subsection 3.2.1, which uses the maximal charging speed to estimate the time needed to charge the amount of energy consumed along the edge, we suggest, that using a wider palette of charging speeds resulting in several different criteria functions $\boldsymbol{\omega}'$, could lead to the better overall quality of retrieved plans.

Influence of POI Scheduling to Performance of CFP-POI Algorithm

Since one of the contributions of this thesis is the integration of scheduling POIs into the planning process, we evaluate its influence on the performance of the CFP-POI algorithm. For this purpose, we compare runtimes of the CFP-POI algorithm with and without scheduling of POIs on the set of 100 queries between locations in regions of northern Germany and central Italy, same as in the case of experiments in Subsection 4.3.2. The initial value of the SoC is set to $\beta_s = 90\ \%$, the minimal destination SoC is set to $\beta_t = 10\ \%$, the duration period of each POI stop is set to $d_{\text{POI}} = 30$ min, and the frequency of POI stops is set to $f_{\text{POI}} = 4$ h.

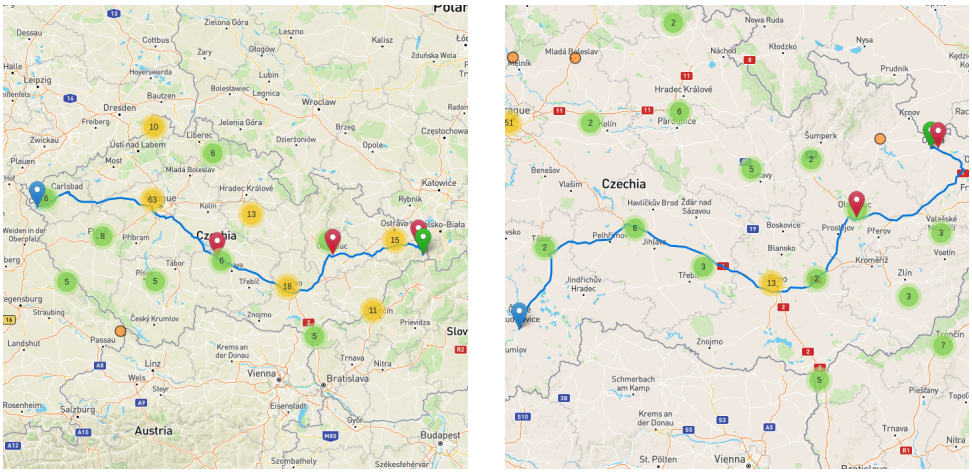
	POIs scheduled	No POIs scheduled
Query time	7.19 s (±0.73)	7.70 s (±0.79)
CFP-POI runtime	2.68 s (±0.92)	3.20 s (±0.89)
Plan time	16.39 h	16.22 h
Charging time	167 min	154 min

Table 4.5: Mean values of the query time and the CFP-POI runtime on 100 queries between locations in the northern Germany and the central Italy with and without scheduling of POIs.

Table 4.5 shows that the presence of scheduling POIs has a positive effect on both query time and the runtime of the CFP-POI algorithm. This is because the constraints for scheduling POIs do not allow the production of labels, which do not satisfy the minimal desired frequency of POI stops and the lower amount of expanded labels leads to lower runtimes of the algorithm. Table 4.5 also indicates that the mean charging time rises by approximately 13 minutes with the application of the POIs scheduling.

Precomputation-Based Heuristic Evaluation

The preprocessing time for the Precomputation-Based heuristic approach introduced in Subsection 3.5.2 may rise to days and even weeks for regions of the continental-scale. The number of charging stations worldwide is still rising, which implies that many queries need to be precomputed using the CFP-POI algorithm with the simple Single-Criterion Search heuristic. Therefore, to save the precomputation time, we evaluate the performance of the Precomputation-Based heuristic on four hand-picked queries, for which



(a) : Path from Cheb to Jablunkov

(b) : Path from České Budějovice to Opava.

Figure 4.11: Paths used for the evaluation of the Precomputation-Based heuristic.

we precompute all needed plans. The critical parameter we use to compare the performance of the CFP-POI algorithm with the Single-Criterion Search heuristic and the Precomputation-Based heuristic is the amount of extracted and dominated labels during the run of the algorithm, which directly influences its final runtime. Chosen queries consist of two pairs of initial and destination locations and two SoC configurations of the initial SoC β_s and the minimal destination SoC β_t for each such pair of locations. The path from Cheb to Jablunkov is shown in Figure 4.11a and the path from České Budějovice to Opava is illustrated in Figure 4.11b.

	$\beta_s = 100, \beta_t = 80$	$\beta_s = 10, \beta_t = 80$
Number of extracted labels	825	1583
Number of dominated labels	8082	39600
CFP-POI runtime	3.91 s	10.18 s

Table 4.6: Characteristics of queries from Cheb to Jablunkov using the CFP-POI algorithm with Single-Criterion Search heuristic.

	$\beta_s = 100, \beta_t = 80$	$\beta_s = 10, \beta_t = 80$
Number of extracted labels	53	79
Number of dominated labels	105	860
CFP-POI runtime	0.07 s	0.22 s

Table 4.7: Characteristics of queries from Cheb to Jablunkov using the CFP-POI algorithm with Precomputation-Based heuristic.

The comparison of values in Tables 4.6 and 4.7 shows that considering the path from České Budějovice to Opava, with the value of the initial SoC $\beta_s = 100$ % and the minimal value of the destination SoC $\beta_t = 80$ %, the num-

	$\beta_s = 100, \beta_t = 80$	$\beta_s = 10, \beta_t = 80$
Number of extracted labels	272	742
Number of dominated labels	829	9268
CFP-POI runtime	0.27 s	2.55 s

Table 4.8: Characteristics of queries from České Budějovice to Opava using the CFP-POI algorithm with Single-Criterion Search heuristic.

	$\beta_s = 90, \beta_t = 10$	$\beta_s = 10, \beta_t = 80$
Number of extracted labels	38	48
Number of dominated labels	43	1680
CFP-POI runtime	0.06 s	0.48 s

Table 4.9: Characteristics of queries from České Budějovice to Opava using the CFP-POI algorithm with Precomputation-Based heuristic.

bers of extracted and dominated labels considerably decrease with the usage of the Precomputation-Based heuristic. Even more significant difference is between queries on the same path with the initial SoC $\beta_s = 100$ % and the minimal value of the destination SoC $\beta_t = 80$ %, where the sum of expanded and dominated queries dropped from 41183 in the case of the Single-Criterion Search heuristic to 939. Such a decrease leads to significantly lower CFP-POI runtime which decreases from 10.18 seconds to 0.22 seconds. Results in Tables 4.8 and 4.9 show similar trend.

Since the Single-Criterion Search heuristic is admissible and even the CFP-POI algorithm is exact, we know that solutions produced by the combination of CFP-POI and the Single-Criterion Search heuristic are optimal. However, it is essential to notice that even though all of our testing queries resulted in entirely identical and hence optimal plans, the Precomputation-Based heuristic is not admissible. It is caused by the heuristic choice of the closest vertices used to estimate the time of the plan. Hence, it may overestimate the total time of the plan, which may result in the resulting plan time being longer than optimum. However, we did not notice this in our experiments.

4.4 Qualitative Experiments comparison of CFP-POI

In the following subsections, we compare plans produced by the CFP-POI algorithm with plans retrieved from the ABRP web application. Furthermore, we compare conditions and overall performance of our solution and results from experiments in Section 4.3, with assumptions, settings and results of experiments provided in [6].

4.4.1 Comparison with the ABRP

We choose two initial and destination location pairs to compare plans generated by our CFP-POI algorithm and plans produced by the ABRP. We set values of the initial SoC β_s to 90 % and values of the destination SoC β_t to 80 %. We also set the minimal value of the SoC $\beta_{\text{safe}} = 6$ %, which stands for the minimal SoC of the EV during the path and serves as a safety margin in the case when the real consumption of the energy is higher than the estimated one. In our experiments, we use the EV model of the Skoda Enyaq iV 80 with consumption parameters set to cold weather, which results in higher consumption. Fortunately, this vehicle is also supported by ABRP. All of our following comparisons use the same EV model.

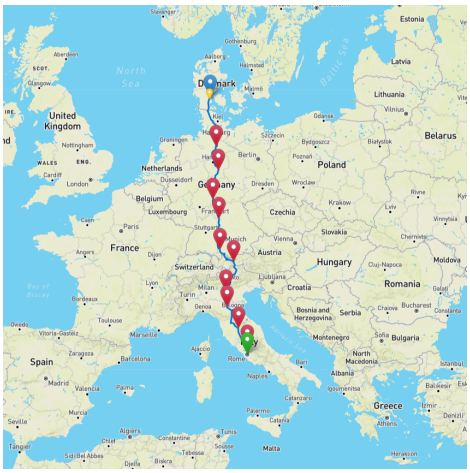
Kolding – Rome

The first pair of locations consists of the city of Kolding, located in southern part of Denmark, and the capital city of Italy, Rome. Examples of routes together with marked charging stops on the path produced by both the CFP-POI and the ABRP are in Figures 4.12a and 4.12b.

	CFP-POI	ABRP
Query time	7.5 s	30.5 s
Plan time	21.67 h	22.22 h
Charging time	4.35 h	5.1 h
Mean charging stop length	26.1 min	27.82 min
Path length	1904 km	1965 km
Charging stops count	10	11
Consumption per 100 km	24.3 kWh	25.8 kWh

Table 4.10: Comparison of plan characteristic between the CFP-POI and the ABRP solutions on the path between Kolding and Rome with $\beta_s = 90$ % and $\beta_t = 80$ %.

Table 4.10 shows that the query time of the CFP-POI algorithm, together with the time for the auxiliary graph construction, is considerably lower than the response time of the ABRP web application. Regarding characteristics of the returned plan, Figures 4.12a and 4.12b show, that the resulting path differs, which declares the comparison of path lengths in Table 4.10 as well. The ABRP planner suggests the path cruising through eastern part of Germany, CFP-POI chooses to pass through western part. Detailed analysis of the query shows that the path returned by the ABRP corresponds to the unconstrained shortest path, optimizing the driving time. Since the auxiliary graph G_{aux} consists of three types of edges optimizing criteria functions $\mathbf{c}$, $\boldsymbol{\omega}$ and, most importantly, edges optimizing the driving time function $\mathbf{d}$, it is also possible to reconstruct the path produced by ABRP in the case of the CFP-POI. However, CFP-POI evaluates that the corridor leading through more consumption-friendly routes results in a lower value of the overall plan time. This is probably one of the factors causing the plan provided by the CFP-POI



(a) : CFP-POI algorithm result with marked charging stations.

(b) : ABRP result with marked charging stations.

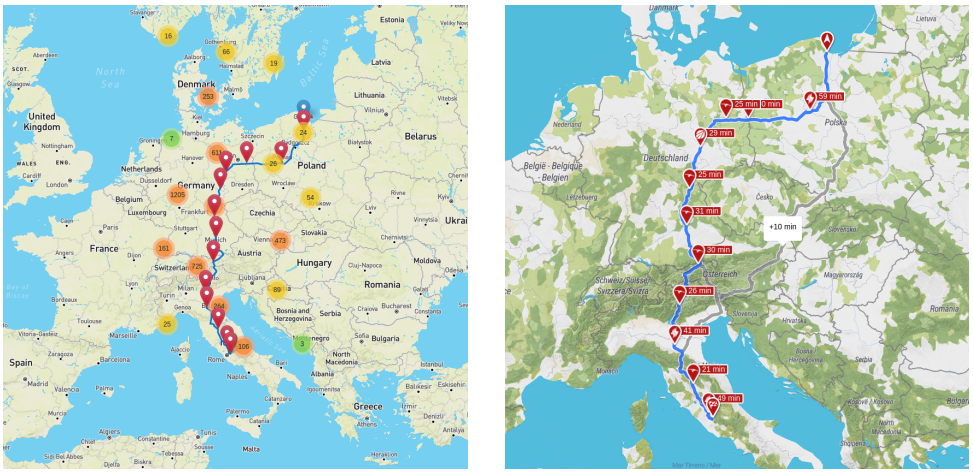
Figure 4.12: Result paths produced by the CFP-POI algorithm and the ABRP between Kolding and Rome.

to be approximately 30 minutes shorter than the one returned by the ABRP. Another causing factor is that even though we use the same EV model (Skoda Enyaq iV 80) in the CFP-POI and the ABRP, we experienced that the consumption and charging model used by ABRP is more pessimistic than ours. This manifests in higher energy consumption and higher charging times, even on identical paths. It is also the case here, where the estimated consumption per 100 km differs by 1.5 kWh, which naturally leads to higher charging time and an amount of needed charging stops. User's discussions on the web show that the conservatism of the ABRP in the case of the consumption estimation is common [53].

Gdansk – Tivoli

The second chosen pair of locations consists of the city of Gdansk in northern Poland and the city of Tivoli, a beautiful historical Italian city 40 kilometers east of Rome.

Figures 4.13a and 4.13b show that contrarily to the previous plan for Kolding and Rome, in the case of Gdansk and Tivoli, both the CFP-POI and the ABRP produced somewhat similar paths passing through the same regions. Even though the length of the path from the ABRP is approximately 30 km longer, we attribute this variance to potential differences in the used map base resulting in different route graphs. The majority of parameters in Table 4.11 comparing plans produced by the CFP-POI and ABRP features by a similar trend as in the case of the query between Kolding and Rome. These are lower query time, plan time, charging time, and consumption per 100 km of the plan produced by the CFP-POI. However, the difference between mean charging stop length is much more significant here, with the difference



(a) : CFP-POI algorithm result with marked charging stations.

(b) : ABRP result with marked charging stations.

Figure 4.13: Result paths produced by the CFP-POI algorithm and the ABRP between Gdansk and Tivoli.

	CFP-POI	ABRP
Query time	5.8 s	36 s
Plan time	23.48 h	24.75 h
Charging time	4.75 h	5.78 h
Mean charging stop length	21.92 min	31.55 min
Path length	2059 km	2090 km
Charging stops count	13	11
Consumption per 100 km	24.5 kWh	25.3 kWh

Table 4.11: Comparison of plan characteristics between the CFP-POI and the ABRP solutions on the path between Gdansk and Tivoli with $\beta_s = 90$ % and $\beta_t = 80$ %.

higher than 10 minutes. This indicates that the CFP-POI uses more the advantage following from the fast charging interval for lower values of the SoC or that the ABRP works with a different charging curve than the one in Figure 3.6a.

4.4.2 Comparison with the Literature

The clear choice of the research paper to compare with is the work of Baum et al. [6], which introduces the CFP algorithm serving as a base for our CFP-POI algorithm. They provide experiments on the road networks of Germany and the road network of entire Europe by producing 1000 random queries with the initial SoC $\beta_s = 100$ % and the destination SoC is set probably to 0 since they do not work with the minimal value of the destination SoC in the paper. Their model considers a battery with 85 kWh capacity, comparable to our 77 kWh battery. Regarding charging stations, they introduce three

different distributions concerning the speed of charging stations. Instead, we use actual data gathered from Open Charge Map with filtered slowest chargers. The closest scenario to ours considers 60 % of chargers with speeds under 22 kWh and 40 % of chargers above this power. Furthermore, Baum et al. model charging functions similar to these in Figure 2.2a by piecewise linear concave functions with six breakpoints, which is a considerable simplification compared to our approach, because we model charging functions $\mathbf{cf}_v$ concerning real user reported charging curves with the granularity of one minute.

The method with the most comparable assumptions to our techniques introduced in [6] is their CFP algorithm with $H_{\boldsymbol{\omega}}$ heuristic, which produces only one new label when expanding extracted label ℓ to one of its neighbors. The expansion is done by selecting the parallel edge, which leads to the best value of the heuristic function $\pi_{\boldsymbol{\omega}}$ in the neighboring vertex. This approach is comparable to our restriction of parallel edges between each pair of vertices in the auxiliary graph to three shortest paths optimizing criteria functions $\mathbf{d}$, $\mathbf{c}$, and $\boldsymbol{\omega}$. At the same time, Baum et al. consider the CFP algorithm with $H_{\boldsymbol{\omega}}$ heuristic as the best trade-off between runtimes of the algorithm and errors from optimal algorithm results without any relaxations. They report the average error from optimum in the overall plan time 0.1 % and 5 % in the worst case.

Baum et al. report that they were able to achieve a mean query time of 38.52 s on 1000 uniformly distributed queries in the region of entire Europe. On the other hand, the worst query time of our CFP-POI algorithm during the evaluation process was 12.99 s in the case of the path from northern Germany to central Italy with the initial SoC $\beta_s = 90$ and the minimal destination SoC $\beta_t = 10$ %. However, values of mean query times are around 8 seconds, which represents a considerable speedup comparing the results presented in [6].

Chapter 5
Conclusion

The essential objective of the thesis was to formalize the problem of planning routes for EVs with integrated charging, and POI stops. Furthermore, the goal was to propose the solution with the main focus on scalability and modeling of real-world conditions regarding the EV consumption and charging process. The consequent work was to implement the proposed solutions and evaluate it on realistic datasets.

From the beginning, we focused on the design of heuristic solutions since we targeted query times in units of seconds on continental-scale regions. The key motivation was to support the user experience of the potential application of methods proposed in the thesis to production. Instead of considering the whole route graph, we introduced techniques for constructing the auxiliary graph, consisting exclusively of initial-destination vertices, the set of heuristically chosen charging stations, and precomputed edges between them. During the construction of the auxiliary graph, we introduced two primary relaxations. First of all, we characterize the energy consumption along precomputed edges in the auxiliary graph by a single value. Due to recuperation, this may lead to a situation, where we consider an edge feasible. However, the EV may run out of energy during the process of passing the edge, as discussed in Subsection 3.2.4. This could be easily solved by saving not only the consumption for each edge, but also the value of minimal SoC to pass the edge, which would consider the elevation profile of the edge and do not permit running out of energy during passing through it. Nevertheless, we do not consider this relaxation crucial in the real-world situations because such running out of energy may happen only in the case of driving with low values of SoC in units of %, which is rarely the case for EV drivers and manufacturers of EVs do not even recommend it because of the deterioration of the battery capacity.

The second relaxation is in using only three edges between each pair of vertices retrieved as the shortest paths optimizing distinct criteria functions instead of saving the set of all non-dominated paths as in the case of [19], or using CH as in the case of [6]. However, as experiments in both papers show, to achieve good scalability, they identically stuck to filtering these non-dominated edges up to lower units, which degenerates the importance of saving all non-dominated paths. Moreover, our approach of using the shortest paths optimizing multiple criteria brings potential flexibility to planning for

EV because it enables us to introduce several criteria functions and, this way, customize the resulting plan. Such an approach could be used, for example, to compute more cost-efficient paths by introducing a criteria function taking into account the prize for consumed energy. Another possible topic for future work is the design of the criteria function, which assigns edges the value regarding the count and attractiveness of POIs in the region, which could contribute to better personification in the field of scheduling POIs.

The CFP-POI algorithm based on the CFP algorithm introduced in [6] forms the core of the proposed solution. It introduces several simplifications resulting from the construction of the auxiliary graph, and it incorporates the possibility of scheduling stops for POIs with defined frequency and length of stops.

Experiments showed that the proposed solution responds to continental-scale queries in the mean query time under 10 seconds, which is a significant improvement compared to related work. Furthermore, we made several observations:

- The initial and the destination SoC influences the query time of the algorithm. The higher need for charging leads to an increase in the query time.
- The construction of the auxiliary graph with lower hundreds of vertices leads to sufficient accuracy.
- The influence of the shortest path edges optimizing the consumption criteria on the solution is negligible. We suggest introducing criteria functions representing different trade-offs between the driving time and the consumption along edges.
- Scheduling of POIs positively affects the query time since it prunes the search space.
- The Precomputation-Based heuristic has the potential to speed up the CFP-POI algorithm significantly.

In conclusion, we showed that proposed heuristic techniques improve scalability considering continental-scale regions while preserving the quality of the plan, which was proven by the comparison with the most widely spread solution for planning routes of EVs ABRP and the current state of the art. We consider the proposed solution a suitable proof of concept for a potential production application. We see the most challenging tasks for future work in the extension of planning more POI categories in the CFP-POI algorithm and elaboration on criteria functions used for constructing the auxiliary graph.

Bibliography

[1] Rami Abousleiman and Osamah Rawashdeh. Energy consumption model of an electric vehicle. In *2015 IEEE transportation electrification conference and expo (ITEC)*, pages 1–5. IEEE, 2015.

[2] Ittai Abraham, Daniel Delling, Amos Fiat, Andrew V Goldberg, and Renato F Werneck. Highway dimension and provably efficient shortest path algorithms. *Journal of the ACM (JACM)*, 63(5):1–26, 2016.

[3] Akhil, Sharma. Learning socket programming in c++. Accessed 14. 5. 2022. URL: `https://www.codingninjas.com/blog/2020/07/06/learning-socket-programming-in-c/`.

[4] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route planning in transportation networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9220 LNCS:19–80, 2016. `arXiv:1504.05140`, `doi:10.1007/978-3-319-49487-6\_2`.

[5] Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, pages 1–10, 2015.

[6] 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, 2019.

[7] Moritz Baum, Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. Energy-optimal routes for electric vehicles. In *Proceedings of the 21st ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 54–63, 2013.

[8] Moritz Baum, Julian Dibbelt, Dorothea Wagner, and Tobias Zündorf. Modeling and engineering constrained shortest path algorithms for battery electric vehicles. *Transportation Science*, 54(6):1571–1600, 2020.

[9] Moritz Baum, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Consumption profiles in route planning for electric vehicles: Theory and applications. In *16th International Symposium on Experimental Algorithms (SEA 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

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

[11] Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. Robust distance queries on massive networks. In *European Symposium on Algorithms*, pages 321–333. Springer, 2014.

[12] Julian Dibbelt, Ben Strasser, and Dorothea Wagner. Customizable contraction hierarchies. *Journal of Experimental Algorithmics (JEA)*, 21:1–49, 2016.

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

[14] Jochen Eisner, Stefan Funke, and Sabine Storandt. Optimal route planning for electric vehicles in large networks. *Proceedings of the National Conference on Artificial Intelligence*, 2:1108–1113, 2011.

[15] Electric Vehicle Database. Skoda enyaq iv 80. Accessed 4. 5. 2022. URL: `https://ev-database.org/car/1280/Skoda-Enyaq-iV-80`.

[16] Enyaq Gorm - simply clever EV. Which is better: Skoda navigation or a better route planner? Accessed 20. 4. 2022. URL: `https://www.youtube.com/watch?v=nN_PJNdqUBA`.

[17] EV-Volumes.com. Global plug-in electric vehicle market share in 2021, by main producer [graph]. Accessed 20. 4. 2022. URL: `https://www.statista.com/statistics/541390/global-sales-of-plug-in-electric-vehicle-manufacturers/`.

[18] Chiara Fiori, Kyoungho Ahn, and Hesham A Rakha. Power-based electric vehicle energy consumption model: Model development and validation. *Applied Energy*, 168:257–268, 2016.

[19] Tomáš Fišer. *Integrated Route and Charging Planning for Electric Vehicles*. PhD thesis, Bachelor's Thesis, Czech Technical University, Prague, Czech Republic, 2017. URL: `https://dspace.cvut.cz/handle/10467/66837`.

[20] Robert W Floyd. Algorithm 97: shortest path. *Communications of the ACM*, 5(6):345, 1962.

[21] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *International Workshop on Experimental and Efficient Algorithms*, pages 319–333. Springer, 2008.

[22] 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.

[23] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, volume 5, pages 156–165. Citeseer, 2005.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2022. URL: `https://www.gurobi.com`.

[25] Gabriel Y Handler and Israel Zang. A dual algorithm for the constrained shortest path problem. *Networks*, 10(4):293–309, 1980.

[26] 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.

[27] Juris Hartmanis. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Review*, 24(1):90, 1982.

[28] Moritz Hilger, Ekkehard Köhler, Rolf H Möhring, and Heiko Schilling. Fast point-to-point shortest path computations with arc-flags. *The Shortest Path Problem: Ninth DIMACS Implementation Challenge*, 74:41–72, 2009.

[29] Hyundai Motor Company. Charge myhyundai. Accessed 26. 4. 2022. URL: `https://chargemyhyundai.com/web/hyundai-cz`.

[30] ID.Furkan. Automatic charge stop planner 21.5. Accessed 20. 4. 2022. URL: `https://www.youtube.com/watch?v=Je7pSBEfCug`.

[31] Iternio Planning AB. A better route planner. Accessed 21. 4. 2022. URL: `https://abetterrouteplanner.com/`.

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

[33] Tim Kieritz, Dennis Luxen, Peter Sanders, and Christian Vetter. Distributed time-dependent contraction hierarchies. In *International Symposium on Experimental Algorithms*, pages 83–93. Springer, 2010.

[34] Ulrich Lauther. An experimental evaluation of point-to-point shortest path calculation on road networks with precalculated edge-flags. *The Shortest Path Problem*, 74:19–39, 2006.

[35] Chensheng Liu, Jing Wu, and Chengnian Long. Joint charging and routing optimization for electric vehicle navigation systems. *IFAC Proceedings Volumes*, 47(3):2106–2111, 2014.

[36] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[37] Ernesto Queirós Vieira Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2):236–245, 1984. URL: https://www.sciencedirect.com/science/article/pii/0377221784900778, doi:https://doi.org/10.1016/0377-2217(84)90077-8.

[38] Jens Maue, Peter Sanders, and Domagoj Matijevic. Goal-directed shortest-path queries using precomputed cluster distances. *Journal of Experimental Algorithmics (JEA)*, 14:3–2, 2010.

[39] Mercedes-Benz. Mercedes me charge. Accessed 20. 4. 2022. URL: https://www.mercedes-benz.cz/passengercars/mercedes-benz-cars/mercedes-me-charge/stage.module.html.

[40] OpenChargeMap contributors. Open charge map. https://openchargemap.org/site, 2014. Accessed 15. 5. 2022.

[41] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. Accessed 15. 5. 2022.

[42] Puente Hills Hyundai. 2022 hyundai ioniq 5 | navigation system. Accessed 26. 4. 2022. URL: https://www.youtube.com/watch?v=MvvAD-neBKc.

[43] Rakesh Patel. A better route planner review – does this ev trip planner help businesses? Accessed 21. 4. 2022. URL: https://www.upperinc.com/reviews/a-better-route-planner-review/.

[44] Rc Obssesive. Skoda enyaq iv 2022 - infotainment system explained. Accessed 22. 4. 2022. URL: https://www.youtube.com/watch?v=-DVXW-beUYQ.

[45] Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Twenty-fifth AAAI conference on artificial intelligence*, 2011.

[46] Frank Schulz, Dorothea Wagner, and Karsten Weihe. Dijkstra's algorithm on-line: An empirical case study from public railroad transport. *ACM J. Exp. Algorithmics*, 5:12–es, dec 2001. doi:10.1145/351827.384254.

[47] SDA Dan Cars. New hyundai infotainment system | multimedia and navigation 2020 | kona ev. Accessed 26. 4. 2022. URL: https://www.youtube.com/watch?v=tTI14R5G12g.

[48] Sabine Storandt. Quick and energy-efficient routes: Computing constrained shortest paths for electric vehicles. *IWCTS 2012 - 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science*, pages 20–25, 2012. doi:10.1145/2442942.2442947.

[49] Sabine Storandt and Stefan Funke. Cruising with a battery-powered vehicle and not getting stranded. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012. URL: http://dblp.uni-trier.de/db/conf/aaai/aaai2012.html#StorandtF12.

[50] Martin Strehler, Sören Merting, and Christian Schwan. Energy-efficient shortest routes for electric and hybrid vehicles. *Transportation Research Part B: Methodological*, 103:111–135, 2017. Green Urban Transportation. URL: https://www.sciencedirect.com/science/article/pii/S0191261516304404, doi:https://doi.org/10.1016/j.trb.2017.03.007.

[51] Tesla. Maps and navigation. [Accessed Apr. 20, 2022]. URL: https://www.tesla.com/ownersmanual/models/en_us/GUID-01F1A582-99D1-4933-B5FB-B2F0203FFE6F.html.

[52] Tesla. Number of tesla vehicles delivered worldwide from 1st quarter 2016 to 1st quarter 2022 (in 1,000 units) [graph]. Accessed 20. 4. 2022. URL: https://www.statista.com/statistics/502208/tesla-quarterly-vehicle-deliveries/.

[53] Tesla Motors Club users. Abrp plan very conservative? Accessed 18. 5. 2022. URL: https://teslamotorsclub.com/tmc/threads/abrp-plan-very-conservative.236279/.

[54] Volkswagen. Electric vehicle route planner. Accessed 20. 4. 2022. URL: https://www.volkswagen.ie/en/electric-cars/charging/electric-vehicle-route-planner.html.

[55] VW ID TALK users. Ev route planner issues. Accessed 20. 4. 2022. URL: https://www.vwidtalk.com/threads/ev-route-planner-issues-stock-system.1588/.

[56] Dorothea Wagner, Thomas Willhalm, and Christos Zaroliagis. Geometric containers for efficient shortest-path computation. *Journal of Experimental Algorithmics (JEA)*, 10:1–3, 2005.

[57] Jiquan Wang, Igo Besselink, and Henk Nijmeijer. Electric vehicle energy consumption modelling and prediction based on road information. *World Electric Vehicle Journal*, 7(3):447–458, 2015.

[58] Zheng Wang and Jon Crowcroft. Quality-of-service routing for supporting multimedia applications. *IEEE Journal on selected areas in communications*, 14(7):1228–1234, 1996.

[59] Wikipedia contributors. Electric car use by country. Accessed 20. 4. 2022. URL: https://en.wikipedia.org/wiki/Electric_car_use_by_country.

[60] Wikipedia contributors. Haversine formula. Accessed 30. 4. 2022. URL: https://en.wikipedia.org/wiki/Haversine_formula.

[61] Wikipedia contributors. Plug-in electric vehicles in europe. Accessed 20. 4. 2022. URL: https://en.wikipedia.org/wiki/Plug-in_electric_vehicles_in_Europe#cite_note-EAFO2020paxM1-3.

[62] Shuai Zhang, Yuvraj Gajpal, SS Appadoo, and MMS Abdulkader. Electric vehicle routing problem with recharging stations for minimizing energy consumption. *International Journal of Production Economics*, 203:404–413, 2018.

[63] Škoda Auto. Where can i find public chargers? Accessed 20. 4. 2022. URL: https://www.skoda-auto.com/emobility/public-charging#anchor-M27-fb21b2ae.

Appendix A

Evaluation Results

In following tables QT stands for the query time and PT stands for the plan time. In the case when table does not contain 100 queries, it means that the missing queries were found infeasible.

Table A.1: Performance and plan quality statistics for the CFP-POI and the ILP model on Prague and Brno regions with $\beta_s = 90$ % and $\beta_t = 10$ %.

ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
1.	0.57 s	2.03 h	0.96 s	2.03 h
2.	0.6 s	2.46 h	0.63 s	2.46 h
3.	0.54 s	2.72 h	5.13 s	2.67 h
4.	0.43 s	2.23 h	4.78 s	2.23 h
5.	0.6 s	2.7 h	5.2 s	2.63 h
6.	0.45 s	2.26 h	1.36 s	2.26 h
7.	0.24 s	1.79 h	0.33 s	1.79 h
8.	0.35 s	2.06 h	0.62 s	2.06 h
9.	0.38 s	2.77 h	0.57 s	2.85 h
10.	0.31 s	1.83 h	0.46 s	1.83 h
11.	0.54 s	2.07 h	0.85 s	2.07 h
12.	0.49 s	2.82 h	4.81 s	2.75 h
13.	0.31 s	2.17 h	0.68 s	2.17 h
14.	0.41 s	2.29 h	4.72 s	2.29 h
15.	0.35 s	1.62 h	0.6 s	1.62 h
16.	0.52 s	3.09 h	4.84 s	3.03 h
17.	0.61 s	2.14 h	4.88 s	2.14 h
18.	0.59 s	2.45 h	4.91 s	2.51 h
19.	0.66 s	2.62 h	5.48 s	2.7 h
20.	0.32 s	2.43 h	1.04 s	2.43 h
21.	0.28 s	1.97 h	0.63 s	1.97 h
22.	0.39 s	2.6 h	1.01 s	2.6 h
23.	0.33 s	2.31 h	0.73 s	2.31 h
24.	0.46 s	2.34 h	1.21 s	2.29 h

Continued on next page

Table A.1 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
25.	0.49 s	2.2 h	1.32 s	2.2 h
26.	0.45 s	2.19 h	0.9 s	2.19 h
27.	0.58 s	1.97 h	4.8 s	1.97 h
28.	0.37 s	1.9 h	0.85 s	1.9 h
29.	0.39 s	2.06 h	0.72 s	2.06 h
30.	0.46 s	2.47 h	1.06 s	2.46 h
31.	0.46 s	2.45 h	4.82 s	2.45 h
32.	0.36 s	2.48 h	0.65 s	2.46 h
33.	0.46 s	2.09 h	0.61 s	2.09 h
34.	1.23 s	2.19 h	4.84 s	2.19 h
35.	0.49 s	2.27 h	0.95 s	2.27 h
36.	0.44 s	1.92 h	0.71 s	1.92 h
37.	0.33 s	1.9 h	0.46 s	1.9 h
38.	0.24 s	1.43 h	0.35 s	1.43 h
39.	0.52 s	2.46 h	1.28 s	2.42 h
40.	0.37 s	1.77 h	4.53 s	1.77 h
41.	0.37 s	1.63 h	0.5 s	1.63 h
42.	0.51 s	2.87 h	5.8 s	2.79 h
43.	0.49 s	2.35 h	1.08 s	2.35 h
44.	0.46 s	2.66 h	0.68 s	2.66 h
45.	0.34 s	2.08 h	0.44 s	2.08 h
46.	0.43 s	2.29 h	0.97 s	2.29 h
47.	0.42 s	1.74 h	0.55 s	1.74 h
48.	0.43 s	2.24 h	0.88 s	2.24 h
49.	0.46 s	2.41 h	0.83 s	2.41 h
50.	0.54 s	2.09 h	0.86 s	2.09 h
51.	0.48 s	2.73 h	4.81 s	2.8 h
52.	0.54 s	2.26 h	5.0 s	2.26 h
53.	0.35 s	1.96 h	0.67 s	1.96 h
54.	0.5 s	1.9 h	1.07 s	1.9 h
55.	0.4 s	2.33 h	0.93 s	2.33 h
56.	0.42 s	2.18 h	0.66 s	2.18 h
57.	0.37 s	2.12 h	0.93 s	2.12 h
58.	0.48 s	2.64 h	4.84 s	2.71 h
59.	1.29 s	2.37 h	1.06 s	2.37 h
60.	0.29 s	2.3 h	0.61 s	2.3 h
61.	0.45 s	2.35 h	4.77 s	2.35 h
62.	0.39 s	1.97 h	0.55 s	1.97 h
63.	0.6 s	2.13 h	1.0 s	2.13 h
64.	0.52 s	2.6 h	4.93 s	2.55 h
65.	0.28 s	1.64 h	0.47 s	1.64 h
66.	0.47 s	2.03 h	0.88 s	2.03 h

Continued on next page

Table A.1 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
67.	0.75 s	2.51 h	0.76 s	2.51 h
68.	0.5 s	2.79 h	4.71 s	2.78 h
69.	0.27 s	1.96 h	0.48 s	1.96 h
70.	0.45 s	2.35 h	4.75 s	2.35 h
71.	0.45 s	2.1 h	0.61 s	2.1 h
72.	0.25 s	1.56 h	0.37 s	1.56 h
73.	0.34 s	2.31 h	0.79 s	2.31 h
74.	0.44 s	2.24 h	0.73 s	2.24 h
75.	0.42 s	2.2 h	3.83 s	2.2 h
76.	0.42 s	1.98 h	0.77 s	1.98 h
77.	0.45 s	2.63 h	4.04 s	2.63 h
78.	0.44 s	2.05 h	0.73 s	2.05 h
79.	0.41 s	2.24 h	0.91 s	2.24 h
80.	0.51 s	2.87 h	5.25 s	2.8 h
81.	0.35 s	1.59 h	0.57 s	1.59 h
82.	0.35 s	2.63 h	4.63 s	2.58 h
83.	0.31 s	2.01 h	0.54 s	2.01 h
84.	0.63 s	2.76 h	3.14 s	2.76 h
85.	0.57 s	2.22 h	2.64 s	2.22 h
86.	0.37 s	2.29 h	0.64 s	2.29 h
87.	0.49 s	2.42 h	4.73 s	2.36 h
88.	0.57 s	3.15 h	6.79 s	3.07 h
89.	0.3 s	1.99 h	0.67 s	1.99 h
90.	0.32 s	1.83 h	0.64 s	1.83 h
91.	0.29 s	1.88 h	0.39 s	1.88 h
92.	0.45 s	2.48 h	0.86 s	2.46 h
93.	0.47 s	2.33 h	1.29 s	2.33 h
94.	0.39 s	1.9 h	0.69 s	1.9 h
95.	0.4 s	2.05 h	0.62 s	2.05 h
96.	0.52 s	2.07 h	4.85 s	2.07 h
97.	0.35 s	2.23 h	0.89 s	2.23 h
98.	0.47 s	2.04 h	0.68 s	2.04 h
99.	0.43 s	2.41 h	1.81 s	2.41 h
100.	0.53 s	2.07 h	4.85 s	2.07 h

Table A.2: Performance and plan quality statistics for the CFP-POI and the ILP model on regions of western and eastern Czech republic with $\beta_s = 90$ % and $\beta_t = 10$ %.

ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
1.	0.86 s	5.58 h	7.54 s	5.42 h
2.	0.77 s	5.11 h	7.22 s	4.96 h
3.	1.27 s	6.12 h	7.86 s	5.95 h
4.	0.81 s	5.33 h	8.55 s	5.16 h
5.	0.79 s	5.44 h	6.8 s	5.29 h
6.	1.45 s	5.28 h	6.19 s	5.14 h
7.	0.81 s	6.51 h	8.7 s	6.34 h
8.	0.85 s	5.89 h	8.91 s	5.72 h
9.	1.34 s	5.98 h	8.99 s	5.81 h
10.	0.77 s	5.83 h	14.16 s	5.66 h
11.	1.34 s	5.63 h	8.5 s	5.46 h
12.	0.75 s	5.73 h	9.04 s	5.56 h
13.	0.7 s	5.5 h	6.53 s	5.33 h
14.	0.75 s	5.85 h	10.11 s	5.68 h
15.	0.69 s	5.56 h	11.9 s	5.43 h
16.	0.74 s	4.91 h	6.14 s	4.76 h
17.	0.67 s	5.44 h	7.78 s	5.29 h
18.	0.82 s	6.29 h	7.81 s	6.13 h
19.	1.29 s	5.92 h	14.28 s	5.76 h
20.	1.89 s	5.83 h	11.8 s	5.66 h
21.	0.77 s	5.56 h	13.31 s	5.39 h
22.	1.0 s	5.19 h	11.91 s	5.02 h
23.	0.73 s	5.72 h	8.7 s	5.55 h
24.	1.57 s	5.84 h	10.84 s	5.67 h
25.	0.8 s	6.79 h	7.33 s	6.62 h
26.	0.93 s	5.98 h	9.54 s	5.81 h
27.	0.85 s	5.63 h	10.48 s	5.46 h
28.	1.72 s	5.4 h	8.05 s	5.26 h
29.	1.69 s	6.16 h	7.88 s	5.99 h
30.	0.87 s	6.02 h	8.28 s	5.85 h
31.	0.73 s	5.53 h	6.53 s	5.36 h
32.	0.79 s	5.57 h	6.92 s	5.42 h
33.	0.89 s	5.97 h	14.32 s	5.83 h
34.	1.48 s	5.86 h	10.09 s	5.69 h
35.	0.86 s	5.81 h	9.57 s	5.64 h
36.	0.81 s	5.83 h	9.51 s	5.66 h
37.	0.77 s	5.25 h	6.52 s	5.1 h
38.	0.76 s	5.21 h	9.78 s	5.04 h
39.	0.8 s	5.58 h	13.86 s	5.41 h
40.	1.47 s	6.15 h	6.63 s	5.99 h

Continued on next page

Table A.2 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
41.	1.58 s	5.3 h	9.01 s	5.14 h
42.	0.96 s	5.4 h	13.01 s	5.23 h
43.	0.86 s	5.84 h	8.27 s	5.69 h
44.	0.67 s	5.24 h	6.75 s	5.09 h
45.	0.71 s	5.5 h	7.0 s	5.34 h
46.	1.4 s	5.96 h	7.72 s	5.8 h
47.	1.48 s	6.05 h	13.83 s	5.88 h
48.	0.76 s	5.5 h	7.66 s	5.36 h
49.	0.67 s	4.92 h	5.97 s	4.78 h
50.	1.43 s	6.14 h	6.95 s	5.97 h
51.	0.85 s	5.66 h	12.05 s	5.49 h
52.	1.35 s	5.46 h	7.68 s	5.31 h
53.	0.7 s	5.37 h	8.46 s	5.2 h
54.	1.31 s	5.77 h	7.86 s	5.6 h
55.	0.71 s	5.63 h	7.71 s	5.47 h
56.	1.24 s	5.51 h	7.5 s	5.34 h
57.	1.48 s	5.35 h	8.08 s	5.22 h
58.	0.77 s	5.86 h	10.4 s	5.69 h
59.	1.41 s	6.77 h	9.49 s	6.61 h
60.	1.43 s	6.72 h	12.96 s	6.62 h
61.	0.71 s	5.36 h	6.75 s	5.19 h
62.	0.81 s	5.84 h	11.99 s	5.67 h
63.	0.71 s	5.48 h	8.42 s	5.4 h
64.	0.79 s	6.1 h	9.82 s	5.95 h
65.	0.77 s	5.66 h	8.61 s	5.49 h
66.	0.73 s	5.21 h	6.3 s	5.04 h
67.	1.37 s	6.98 h	9.94 s	6.81 h
68.	0.71 s	5.17 h	7.26 s	5.01 h
69.	1.49 s	5.44 h	8.08 s	5.27 h
70.	1.26 s	5.67 h	9.32 s	5.5 h
71.	0.85 s	5.8 h	8.86 s	5.63 h
72.	0.84 s	5.79 h	7.01 s	5.62 h
73.	1.35 s	5.82 h	7.31 s	5.65 h
74.	1.34 s	5.78 h	6.77 s	5.61 h
75.	0.7 s	5.75 h	7.31 s	5.59 h
76.	0.78 s	5.68 h	9.57 s	5.51 h
77.	1.28 s	5.17 h	7.18 s	5.0 h
78.	0.76 s	5.77 h	7.25 s	5.6 h
79.	0.69 s	5.85 h	9.47 s	5.68 h
80.	0.73 s	5.55 h	11.99 s	5.38 h
81.	0.69 s	5.42 h	8.55 s	5.26 h
82.	1.32 s	5.71 h	8.74 s	5.56 h

Continued on next page

Table A.2 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
83.	0.77 s	6.6 h	9.13 s	6.43 h
84.	0.77 s	5.5 h	8.28 s	5.33 h
85.	0.77 s	5.59 h	7.72 s	5.43 h
86.	0.69 s	5.65 h	9.98 s	5.48 h
87.	0.78 s	5.25 h	8.87 s	5.09 h
88.	0.75 s	5.48 h	8.82 s	5.31 h
89.	1.42 s	6.32 h	8.4 s	6.17 h
90.	0.79 s	5.53 h	9.76 s	5.36 h
91.	1.34 s	5.51 h	8.9 s	5.34 h
92.	1.24 s	5.89 h	8.41 s	5.72 h
93.	0.8 s	5.86 h	14.17 s	5.7 h
94.	0.84 s	6.0 h	10.87 s	5.91 h
95.	1.35 s	5.96 h	7.92 s	5.79 h
96.	0.7 s	5.88 h	10.02 s	5.76 h
97.	0.74 s	5.75 h	13.43 s	5.59 h
98.	0.72 s	5.44 h	8.38 s	5.27 h
99.	0.84 s	5.88 h	8.31 s	5.71 h
100.	0.78 s	6.09 h	14.16 s	5.92 h

Table A.3: Performance and plan quality statistics for the CFP-POI and the ILP model on regions of western and eastern Czech republic with $\beta_s = 10$ % and $\beta_t = 80$ %.

ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
1.	2.64 s	7.57 h	51.34 s	7.2 h
2.	3.67 s	7.59 h	57.68 s	7.43 h
3.	1.7 s	8.64 h	47.88 s	8.21 h
4.	2.35 s	7.15 h	50.21 s	6.78 h
5.	1.64 s	7.55 h	57.78 s	7.3 h
6.	2.23 s	7.73 h	57.42 s	7.65 h
7.	4.15 s	8.63 h	57.61 s	8.42 h
8.	1.79 s	7.94 h	57.88 s	7.54 h
9.	1.95 s	8.41 h	57.58 s	8.24 h
10.	2.2 s	7.86 h	37.34 s	7.49 h
11.	1.93 s	8.05 h	57.69 s	7.62 h
12.	1.46 s	7.77 h	37.52 s	7.34 h
13.	1.19 s	8.0 h	57.64 s	7.58 h
14.	2.11 s	7.86 h	41.05 s	7.49 h
15.	2.06 s	7.4 h	37.66 s	6.97 h
16.	1.51 s	7.36 h	37.52 s	7.18 h

Continued on next page

Table A.3 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
17.	2.66 s	7.68 h	54.01 s	7.25 h
18.	3.51 s	8.45 h	57.58 s	8.03 h
19.	1.74 s	8.31 h	57.78 s	7.89 h
20.	3.23 s	7.91 h	57.66 s	7.52 h
21.	2.03 s	7.62 h	37.4 s	7.2 h
22.	1.37 s	7.51 h	37.36 s	7.08 h
23.	2.16 s	7.59 h	37.49 s	7.16 h
24.	3.0 s	7.83 h	57.64 s	7.46 h
25.	2.49 s	9.0 h	48.01 s	8.58 h
26.	2.7 s	8.04 h	57.61 s	7.65 h
27.	2.16 s	7.92 h	57.59 s	7.49 h
28.	2.13 s	7.46 h	57.83 s	7.2 h
29.	2.45 s	8.47 h	57.83 s	8.1 h
30.	1.87 s	8.14 h	47.11 s	7.72 h
31.	1.5 s	7.87 h	57.64 s	7.5 h
32.	2.59 s	7.69 h	57.47 s	7.55 h
33.	1.5 s	8.01 h	57.71 s	7.67 h
34.	2.41 s	7.82 h	57.68 s	7.4 h
35.	1.99 s	7.78 h	37.52 s	7.39 h
36.	2.65 s	7.94 h	49.27 s	7.51 h
37.	2.87 s	7.68 h	28.28 s	7.32 h
38.	1.17 s	7.48 h	37.43 s	7.05 h
39.	1.35 s	7.5 h	28.35 s	7.07 h
40.	2.86 s	8.34 h	28.31 s	7.92 h
41.	2.64 s	7.63 h	37.73 s	7.2 h
42.	2.2 s	7.61 h	57.5 s	7.19 h
43.	3.08 s	8.19 h	37.37 s	7.8 h
44.	1.72 s	7.64 h	57.48 s	7.21 h
45.	1.38 s	8.18 h	37.74 s	7.57 h
46.	2.24 s	7.95 h	37.48 s	7.52 h
47.	2.18 s	8.08 h	57.6 s	7.65 h
48.	3.54 s	7.82 h	57.89 s	7.53 h
49.	2.11 s	7.49 h	37.49 s	7.28 h
50.	1.79 s	8.15 h	38.72 s	7.76 h
51.	1.59 s	7.66 h	57.72 s	7.28 h
52.	2.18 s	7.42 h	46.06 s	6.99 h
53.	1.49 s	7.85 h	37.5 s	7.42 h
54.	2.03 s	7.96 h	49.88 s	7.53 h
55.	2.53 s	7.76 h	57.12 s	7.34 h
56.	2.18 s	8.19 h	54.5 s	7.8 h
57.	1.34 s	7.22 h	57.43 s	7.1 h
58.	4.19 s	7.91 h	57.54 s	7.59 h

Continued on next page

Table A.3 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
59.	4.42 s	8.98 h	57.49 s	8.56 h
60.	2.53 s	8.78 h	57.55 s	8.41 h
61.	2.56 s	7.8 h	57.87 s	7.37 h
62.	1.5 s	7.98 h	57.64 s	7.55 h
63.	2.42 s	7.43 h	57.6 s	7.2 h
64.	1.66 s	8.1 h	57.56 s	7.74 h
65.	2.35 s	7.85 h	40.81 s	7.43 h
66.	3.17 s	7.68 h	57.77 s	7.41 h
67.	2.33 s	9.39 h	38.82 s	9.08 h
68.	1.93 s	7.62 h	57.39 s	7.53 h
69.	2.14 s	7.48 h	57.5 s	7.05 h
70.	1.92 s	7.74 h	52.16 s	7.31 h
71.	2.41 s	7.87 h	37.46 s	7.45 h
72.	2.53 s	7.87 h	57.56 s	7.45 h
73.	2.59 s	7.84 h	37.53 s	7.47 h
74.	2.53 s	7.82 h	57.51 s	7.42 h
75.	2.59 s	7.95 h	54.77 s	7.56 h
76.	1.42 s	7.72 h	37.38 s	7.39 h
77.	2.92 s	7.74 h	37.54 s	7.33 h
78.	2.24 s	7.84 h	40.66 s	7.42 h
79.	1.32 s	7.72 h	37.28 s	7.29 h
80.	2.25 s	7.54 h	46.16 s	7.12 h
81.	2.18 s	7.68 h	57.49 s	7.26 h
82.	1.56 s	7.63 h	57.46 s	7.25 h
83.	1.68 s	8.85 h	57.51 s	8.42 h
84.	2.34 s	7.45 h	37.32 s	7.02 h
85.	2.35 s	7.4 h	37.44 s	6.98 h
86.	1.54 s	7.92 h	37.41 s	7.5 h
87.	1.26 s	7.96 h	37.41 s	7.36 h
88.	1.79 s	7.64 h	46.63 s	7.21 h
89.	2.54 s	8.47 h	57.61 s	8.04 h
90.	1.91 s	7.38 h	37.67 s	7.0 h
91.	1.67 s	7.71 h	37.69 s	7.29 h
92.	1.3 s	8.22 h	28.24 s	7.79 h
93.	2.76 s	8.17 h	57.55 s	7.84 h
94.	2.56 s	8.07 h	57.55 s	7.65 h
95.	1.41 s	7.85 h	28.36 s	7.42 h
96.	2.33 s	8.02 h	37.35 s	7.63 h
97.	1.55 s	7.92 h	57.49 s	7.5 h
98.	2.02 s	7.52 h	57.59 s	7.1 h
99.	2.75 s	7.89 h	45.28 s	7.47 h
100.	1.5 s	8.39 h	57.58 s	8.04 h

Table A.4: Performance and plan quality statistics for the CFP-POI and the ILP model on regions of western and eastern Czech republic with $\beta_s = 100$ % and $\beta_t = 80$ %.

ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
1.	0.98 s	6.48 h	20.05 s	6.22 h
2.	0.83 s	6.11 h	28.19 s	6.04 h
3.	0.94 s	7.15 h	14.05 s	6.9 h
4.	0.86 s	6.22 h	20.03 s	5.96 h
5.	2.2 s	6.6 h	28.17 s	6.37 h
6.	0.94 s	6.26 h	13.97 s	6.31 h
7.	1.91 s	7.58 h	20.23 s	7.33 h
8.	0.99 s	6.86 h	28.27 s	6.62 h
9.	1.16 s	7.0 h	28.17 s	6.75 h
10.	1.15 s	6.78 h	20.09 s	6.56 h
11.	1.09 s	6.65 h	20.03 s	6.4 h
12.	1.04 s	6.69 h	20.16 s	6.43 h
13.	0.75 s	6.55 h	28.05 s	6.29 h
14.	0.97 s	6.78 h	20.16 s	6.53 h
15.	0.87 s	6.48 h	20.09 s	6.26 h
16.	0.77 s	6.15 h	20.03 s	5.84 h
17.	0.8 s	6.44 h	20.09 s	6.28 h
18.	1.12 s	7.35 h	28.25 s	7.17 h
19.	1.08 s	6.95 h	28.24 s	6.7 h
20.	1.16 s	6.79 h	28.25 s	6.54 h
21.	0.99 s	6.46 h	20.15 s	6.21 h
22.	0.84 s	6.06 h	20.03 s	5.8 h
23.	1.36 s	6.58 h	20.03 s	6.38 h
24.	0.86 s	6.74 h	20.39 s	6.54 h
25.	1.34 s	7.83 h	20.2 s	7.58 h
26.	0.93 s	6.96 h	28.5 s	6.71 h
27.	0.86 s	6.53 h	20.16 s	6.28 h
28.	0.98 s	6.38 h	28.11 s	6.27 h
29.	0.94 s	7.25 h	28.12 s	7.0 h
30.	1.42 s	7.05 h	28.23 s	6.8 h
31.	0.8 s	6.54 h	20.09 s	6.29 h
32.	0.9 s	6.79 h	28.17 s	6.64 h
33.	0.95 s	6.85 h	28.27 s	6.6 h
34.	1.22 s	6.73 h	20.1 s	6.48 h
35.	1.09 s	6.71 h	20.14 s	6.46 h
36.	0.95 s	6.86 h	28.11 s	6.6 h
37.	1.2 s	6.48 h	20.13 s	6.29 h

Continued on next page

Table A.4 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
38.	1.02 s	6.07 h	20.42 s	5.81 h
39.	0.91 s	6.48 h	28.28 s	6.23 h
40.	2.15 s	7.14 h	20.14 s	6.89 h
41.	1.52 s	6.18 h	20.0 s	5.92 h
42.	0.99 s	6.42 h	28.14 s	6.17 h
43.	0.73 s	6.92 h	20.07 s	6.67 h
44.	0.97 s	6.23 h	20.03 s	6.01 h
45.	1.14 s	6.52 h	28.15 s	6.29 h
46.	0.86 s	6.87 h	20.23 s	6.62 h
47.	1.13 s	6.96 h	28.25 s	6.76 h
48.	0.75 s	6.49 h	28.18 s	6.47 h
49.	1.03 s	6.07 h	12.49 s	5.81 h
50.	0.94 s	7.05 h	20.22 s	6.79 h
51.	1.04 s	6.55 h	20.22 s	6.3 h
52.	0.79 s	6.51 h	28.05 s	6.27 h
53.	1.25 s	6.39 h	28.07 s	6.17 h
54.	0.76 s	6.85 h	20.04 s	6.6 h
55.	0.82 s	6.68 h	20.12 s	6.43 h
56.	0.75 s	6.52 h	20.06 s	6.27 h
57.	0.75 s	6.54 h	27.91 s	6.39 h
58.	1.63 s	6.89 h	28.16 s	6.71 h
59.	1.55 s	7.85 h	28.15 s	7.6 h
60.	0.88 s	7.78 h	28.22 s	7.59 h
61.	0.83 s	6.37 h	20.05 s	6.17 h
62.	0.86 s	6.74 h	28.11 s	6.49 h
63.	3.72 s	6.63 h	28.28 s	6.45 h
64.	1.71 s	7.14 h	28.22 s	6.91 h
65.	1.03 s	6.73 h	28.12 s	6.48 h
66.	0.9 s	6.21 h	28.08 s	5.95 h
67.	0.97 s	7.87 h	13.98 s	7.67 h
68.	0.82 s	6.35 h	20.03 s	6.1 h
69.	0.9 s	6.33 h	20.12 s	6.13 h
70.	1.58 s	6.7 h	20.19 s	6.45 h
71.	0.9 s	6.79 h	20.17 s	6.55 h
72.	0.86 s	6.78 h	28.18 s	6.53 h
73.	2.7 s	6.8 h	20.03 s	6.55 h
74.	1.17 s	6.82 h	28.16 s	6.56 h
75.	0.84 s	6.87 h	20.17 s	6.62 h
76.	0.94 s	6.57 h	20.18 s	6.32 h
77.	0.87 s	6.21 h	20.07 s	5.95 h
78.	0.9 s	6.75 h	28.17 s	6.5 h
79.	0.93 s	6.76 h	28.16 s	6.56 h

Continued on next page

Table A.4 – continued from previous page

Query ID	CFP-POI QT	CFP-POI PT	ILP QT	ILP PT
80.	1.0 s	6.43 h	28.1 s	6.23 h
81.	0.9 s	6.29 h	20.12 s	6.04 h
82.	1.14 s	6.66 h	28.19 s	6.43 h
83.	1.53 s	7.65 h	28.17 s	7.47 h
84.	1.0 s	6.41 h	20.16 s	6.16 h
85.	0.99 s	6.5 h	28.08 s	6.25 h
86.	0.89 s	6.67 h	20.12 s	6.42 h
87.	0.88 s	6.12 h	20.0 s	5.86 h
88.	0.86 s	6.45 h	20.11 s	6.19 h
89.	0.94 s	7.39 h	20.24 s	7.15 h
90.	0.96 s	6.39 h	28.17 s	6.14 h
91.	0.81 s	6.39 h	20.24 s	6.14 h
92.	0.88 s	6.75 h	20.16 s	6.51 h
93.	0.88 s	6.88 h	28.14 s	6.64 h
94.	0.79 s	7.03 h	28.14 s	6.86 h
95.	0.83 s	6.84 h	20.11 s	6.59 h
96.	0.78 s	6.89 h	20.15 s	6.64 h
97.	0.79 s	6.77 h	20.08 s	6.55 h
98.	1.05 s	6.47 h	28.16 s	6.22 h
99.	0.98 s	6.84 h	20.14 s	6.62 h
100.	0.81 s	7.1 h	28.28 s	6.85 h

Table A.5: Comparison of the CFP-POI performance with the varying initial SoC β_s and the destination SoC β_t using queries between northern Germany and central Italy.

ID	$\beta_s = 90, \beta_t = 10$	$\beta_s = 10, \beta_t = 80$	$\beta_s = 10, \beta_t = 80$
1.	6.98 s	8.72 s	7.92 s
2.	8.4 s	11.89 s	9.71 s
3.	7.23 s	8.49 s	7.0 s
4.	7.83 s	8.1 s	7.56 s
5.	8.04 s	9.94 s	8.96 s
6.	7.23 s	9.3 s	7.88 s
7.	7.23 s	8.62 s	7.58 s
8.	7.84 s	8.53 s	8.38 s
9.	6.7 s	9.36 s	7.87 s
10.	10.12 s	12.12 s	9.36 s
11.	8.8 s	9.45 s	7.47 s
12.	6.89 s	8.91 s	7.73 s
13.	7.86 s	8.55 s	7.54 s

Continued on next page

Table A.5 – continued from previous page

Query ID	$\beta_s = 90, \beta_t = 10$	$\beta_s = 10, \beta_t = 80$	$\beta_s = 90, \beta_t = 80$
14.	7.98 s	9.91 s	8.28 s
15.	6.64 s	8.34 s	6.98 s
16.	9.5 s	12.43 s	9.36 s
17.	8.79 s	9.69 s	8.58 s
18.	7.12 s	9.48 s	7.9 s
19.	7.51 s	9.13 s	7.21 s
20.	8.19 s	10.99 s	8.59 s
21.	8.52 s	12.77 s	9.93 s
22.	9.76 s	10.54 s	7.85 s
23.	7.3 s	8.88 s	9.46 s
24.	6.23 s	8.17 s	6.76 s
25.	7.44 s	7.71 s	8.63 s
26.	7.32 s	10.63 s	9.24 s
27.	7.77 s	8.27 s	8.57 s
28.	7.54 s	9.31 s	8.76 s
29.	7.68 s	9.95 s	9.87 s
30.	7.21 s	9.69 s	7.75 s
31.	8.19 s	9.24 s	9.12 s
32.	7.36 s	8.48 s	7.71 s
33.	7.88 s	11.66 s	8.84 s
34.	8.18 s	9.1 s	8.09 s
35.	7.28 s	9.43 s	6.83 s
36.	8.28 s	10.08 s	9.48 s
37.	8.92 s	9.85 s	10.43 s
38.	7.76 s	7.78 s	6.94 s
39.	6.5 s	8.15 s	8.57 s
40.	6.9 s	8.74 s	7.73 s
41.	7.97 s	9.21 s	7.81 s
42.	6.79 s	9.88 s	7.72 s
43.	7.16 s	8.82 s	7.19 s
44.	8.26 s	11.92 s	7.85 s
45.	7.81 s	10.67 s	7.61 s
46.	7.86 s	9.77 s	7.93 s
47.	8.04 s	9.73 s	7.88 s
48.	7.15 s	11.7 s	10.03 s
49.	9.54 s	8.39 s	8.42 s
50.	8.57 s	12.46 s	10.59 s
51.	7.37 s	10.16 s	7.14 s
52.	6.87 s	9.95 s	7.98 s
53.	7.5 s	8.08 s	7.26 s
54.	7.23 s	9.37 s	8.93 s
55.	8.89 s	9.07 s	8.93 s

Continued on next page

Table A.5 – continued from previous page

Query ID	$\beta_s = 90, \beta_t = 10$	$\beta_s = 10, \beta_t = 80$	$\beta_s = 90, \beta_t = 80$
56.	9.96 s	10.5 s	8.12 s
57.	7.59 s	11.21 s	10.06 s
58.	7.29 s	10.06 s	8.27 s
59.	7.37 s	8.36 s	8.33 s
60.	6.79 s	8.66 s	7.14 s
61.	8.17 s	11.18 s	9.42 s
62.	8.41 s	8.75 s	8.48 s
63.	7.76 s	10.64 s	8.22 s
64.	7.71 s	10.84 s	8.42 s
65.	6.92 s	12.11 s	11.53 s
66.	11.71 s	10.77 s	8.35 s
67.	6.78 s	11.02 s	9.4 s
68.	9.6 s	8.82 s	8.47 s
69.	9.34 s	9.99 s	8.38 s
70.	7.45 s	12.1 s	9.13 s
71.	8.06 s	8.39 s	7.4 s
72.	12.99 s	11.39 s	7.76 s
73.	7.29 s	8.09 s	8.6 s
74.	7.66 s	9.32 s	9.81 s
75.	7.13 s	9.27 s	7.06 s
76.	7.18 s	9.07 s	7.29 s
77.	7.57 s	9.65 s	8.5 s
78.	7.02 s	8.78 s	7.54 s
79.	9.03 s	7.73 s	7.39 s
80.	6.89 s	8.43 s	7.52 s
81.	7.38 s	8.02 s	8.37 s
82.	7.36 s	9.44 s	7.54 s
83.	8.03 s	8.63 s	9.26 s
84.	6.84 s	11.22 s	10.19 s
85.	7.26 s	7.99 s	7.9 s
86.	9.48 s	10.17 s	10.18 s
87.	7.41 s	11.85 s	9.48 s
88.	7.46 s	10.81 s	8.18 s
89.	6.54 s	9.29 s	9.75 s
90.	8.08 s	7.45 s	7.24 s
91.	7.99 s	8.86 s	8.7 s

Table A.6: Comparison of the plan quality considering different sizes of the auxiliary graph using queries between northern Poland and central Switzerland.

ID	20 per corridor	150 per corridor	350 per corridor
1.	14.82 h	14.81 h	14.81 h
2.	11.0 h	10.98 h	10.98 h
3.	11.24 h	11.14 h	11.14 h
4.	12.55 h	12.48 h	12.48 h
5.	12.85 h	12.85 h	12.85 h
6.	13.11 h	13.04 h	13.04 h
7.	13.13 h	13.11 h	13.11 h
8.	12.63 h	12.63 h	12.63 h
9.	12.74 h	12.66 h	12.66 h
10.	11.18 h	11.14 h	11.14 h
11.	13.05 h	13.02 h	13.02 h
12.	11.78 h	11.74 h	11.74 h
13.	16.36 h	16.33 h	16.33 h
14.	12.45 h	12.43 h	12.43 h
15.	14.41 h	14.37 h	14.37 h
16.	11.52 h	11.44 h	11.44 h
17.	15.58 h	15.53 h	15.53 h
18.	12.77 h	12.69 h	12.69 h
19.	13.06 h	13.01 h	13.01 h
20.	11.9 h	11.89 h	11.89 h
21.	14.14 h	14.06 h	14.06 h
22.	11.24 h	11.22 h	11.22 h
23.	11.81 h	11.78 h	11.78 h
24.	14.07 h	14.03 h	14.03 h
25.	11.8 h	11.74 h	11.74 h
26.	13.93 h	13.93 h	13.93 h
27.	9.99 h	9.97 h	9.97 h
28.	11.5 h	11.42 h	11.42 h
29.	10.8 h	10.78 h	10.78 h
30.	10.56 h	10.5 h	10.5 h
31.	16.1 h	16.1 h	16.1 h
32.	13.4 h	13.4 h	13.4 h
33.	13.24 h	13.22 h	13.22 h
34.	11.51 h	11.48 h	11.48 h
35.	10.89 h	10.88 h	10.88 h
36.	13.15 h	13.12 h	13.12 h
37.	11.34 h	11.34 h	11.34 h
38.	11.45 h	11.43 h	11.43 h
39.	12.45 h	12.44 h	12.44 h
40.	12.1 h	12.07 h	12.07 h

Continued on next page

Table A.6 – continued from previous page

ID	20 per corridor	150 per corridor	350 per corridor
41.	11.14 h	11.14 h	11.14 h
42.	12.78 h	12.71 h	12.71 h
43.	12.83 h	12.83 h	12.83 h
44.	12.36 h	12.28 h	12.28 h
45.	12.56 h	12.49 h	12.49 h
46.	11.2 h	11.17 h	11.17 h
47.	11.56 h	11.55 h	11.55 h
48.	12.18 h	12.09 h	12.09 h
49.	11.85 h	11.82 h	11.82 h
50.	13.78 h	13.78 h	13.78 h
51.	12.88 h	12.88 h	12.88 h
52.	16.53 h	16.53 h	16.53 h
53.	13.04 h	12.99 h	12.99 h
54.	11.5 h	11.48 h	11.48 h
55.	11.88 h	11.83 h	11.83 h
56.	13.16 h	13.08 h	13.08 h
57.	10.8 h	10.75 h	10.75 h
58.	13.69 h	13.65 h	13.65 h
59.	14.58 h	14.52 h	14.52 h
60.	10.53 h	10.46 h	10.46 h
61.	11.64 h	11.62 h	11.62 h
62.	12.64 h	12.63 h	12.63 h
63.	15.13 h	15.08 h	15.08 h
64.	13.13 h	13.12 h	13.12 h
65.	13.3 h	13.27 h	13.27 h
66.	13.59 h	13.57 h	13.57 h
67.	11.14 h	11.14 h	11.14 h
68.	11.02 h	10.96 h	10.96 h
69.	11.46 h	11.39 h	11.39 h
70.	13.08 h	13.06 h	13.06 h
71.	10.5 h	10.44 h	10.44 h
72.	12.87 h	12.82 h	12.82 h
73.	14.58 h	14.56 h	14.47 h
74.	14.16 h	14.14 h	14.14 h
75.	12.08 h	12.03 h	12.03 h
76.	14.32 h	14.2 h	14.2 h
77.	10.94 h	10.91 h	10.91 h
78.	11.38 h	11.36 h	11.36 h
79.	14.02 h	14.02 h	14.02 h
80.	14.36 h	14.32 h	14.32 h
81.	12.66 h	12.62 h	12.62 h
82.	11.76 h	11.7 h	11.7 h

Continued on next page

Table A.6 – continued from previous page

ID	20 per corridor	150 per corridor	350 per corridor
83.	13.23 h	13.2 h	13.2 h
84.	11.83 h	11.77 h	11.77 h
85.	10.52 h	10.49 h	10.49 h
86.	12.08 h	12.03 h	12.03 h
87.	11.98 h	11.96 h	11.96 h
88.	14.13 h	14.05 h	14.05 h
89.	13.49 h	13.45 h	13.45 h
90.	13.43 h	13.43 h	13.43 h
91.	10.84 h	10.76 h	10.76 h
92.	14.88 h	14.82 h	14.82 h
93.	14.21 h	14.21 h	14.21 h
94.	11.41 h	11.33 h	11.33 h
95.	12.96 h	12.86 h	12.86 h
96.	11.87 h	11.84 h	11.84 h
97.	14.28 h	14.26 h	14.26 h
98.	13.48 h	13.45 h	13.45 h
99.	10.44 h	10.44 h	10.44 h

Table A.7: Comparison of the plan quality considering presence of edges optimizing different criteria using queries between central Denmark and northern Croatia.

ID	All edges	No consumption edges	Only time edges
1.	20.12 h	20.12 h	20.16 h
2.	20.02 h	20.02 h	20.11 h
3.	18.96 h	18.96 h	19.0 h
4.	19.42 h	19.42 h	19.46 h
5.	19.13 h	19.13 h	19.18 h
6.	19.08 h	19.08 h	19.12 h
7.	21.25 h	21.25 h	21.29 h
8.	19.18 h	19.18 h	19.25 h
9.	19.55 h	19.55 h	19.59 h
10.	19.09 h	19.09 h	19.13 h
11.	19.12 h	19.12 h	19.16 h
12.	19.99 h	19.99 h	20.03 h
13.	20.01 h	20.01 h	20.09 h
14.	19.4 h	19.4 h	19.48 h
15.	19.65 h	19.65 h	19.69 h
16.	19.2 h	19.2 h	19.24 h
17.	19.41 h	19.41 h	19.44 h

Continued on next page

Table A.7 – continued from previous page

ID	All edges	No consumption edges	Only time edges
18.	19.27 h	19.27 h	19.4 h
19.	18.94 h	18.94 h	18.98 h
20.	19.1 h	19.1 h	19.14 h
21.	19.35 h	19.35 h	19.38 h
22.	19.13 h	19.13 h	19.17 h
23.	20.06 h	20.06 h	20.1 h
24.	19.26 h	19.26 h	19.33 h
25.	19.05 h	19.05 h	19.09 h
26.	19.83 h	19.83 h	19.86 h
27.	19.19 h	19.19 h	19.23 h
28.	19.95 h	19.95 h	19.99 h
29.	19.2 h	19.2 h	19.28 h
30.	19.66 h	19.66 h	19.7 h
31.	18.42 h	18.42 h	18.46 h
32.	19.62 h	19.62 h	19.66 h
33.	20.02 h	20.02 h	20.06 h
34.	19.04 h	19.04 h	19.09 h
35.	18.79 h	18.79 h	18.83 h
36.	19.45 h	19.45 h	19.55 h
37.	18.99 h	18.99 h	19.03 h
38.	19.71 h	19.71 h	19.75 h
39.	19.09 h	19.09 h	19.14 h
40.	19.79 h	19.79 h	19.83 h
41.	19.49 h	19.49 h	19.53 h
42.	18.82 h	18.82 h	18.86 h
43.	18.54 h	18.54 h	18.58 h
44.	19.43 h	19.43 h	19.48 h
45.	19.09 h	19.09 h	19.13 h
46.	19.58 h	19.58 h	19.62 h
47.	20.08 h	20.08 h	20.21 h
48.	18.56 h	18.56 h	18.6 h
49.	20.15 h	20.15 h	20.19 h
50.	18.44 h	18.44 h	18.47 h
51.	19.0 h	19.0 h	19.03 h
52.	19.94 h	19.94 h	19.98 h
53.	19.35 h	19.35 h	19.39 h
54.	19.19 h	19.19 h	19.23 h
55.	19.73 h	19.73 h	19.77 h
56.	18.93 h	18.93 h	18.97 h
57.	19.01 h	19.01 h	19.23 h
58.	19.37 h	19.37 h	19.41 h
59.	19.42 h	19.42 h	19.46 h

Continued on next page

Table A.7 – continued from previous page

ID	All edges	No consumption edges	Only time edges
60.	19.89 h	19.89 h	19.93 h
61.	19.31 h	19.31 h	19.35 h
62.	19.01 h	19.01 h	19.14 h
63.	18.28 h	18.28 h	18.31 h
64.	19.21 h	19.21 h	19.26 h
65.	19.5 h	19.5 h	19.54 h
66.	20.08 h	20.08 h	20.12 h
67.	19.79 h	19.79 h	19.83 h
68.	19.27 h	19.27 h	19.31 h
69.	19.1 h	19.1 h	19.15 h
70.	19.65 h	19.65 h	19.69 h
71.	19.89 h	19.89 h	19.93 h
72.	19.09 h	19.09 h	19.13 h
73.	20.11 h	20.11 h	20.14 h
74.	18.37 h	18.37 h	18.41 h
75.	20.07 h	20.07 h	20.14 h
76.	19.7 h	19.7 h	19.74 h
77.	19.37 h	19.37 h	19.43 h
78.	19.0 h	19.0 h	19.04 h
79.	18.59 h	18.59 h	18.63 h
80.	19.51 h	19.51 h	19.55 h
81.	19.44 h	19.44 h	19.48 h
82.	18.69 h	18.69 h	18.73 h
83.	19.03 h	19.03 h	19.07 h
84.	19.0 h	19.0 h	19.08 h
85.	18.94 h	18.94 h	18.98 h
86.	18.88 h	18.88 h	18.96 h
87.	19.96 h	19.96 h	19.99 h
88.	20.26 h	20.26 h	20.3 h
89.	19.46 h	19.46 h	19.5 h
90.	19.56 h	19.56 h	19.6 h
91.	18.46 h	18.46 h	18.5 h
92.	20.0 h	20.0 h	20.08 h
93.	19.56 h	19.56 h	19.6 h
94.	19.23 h	19.23 h	19.38 h
95.	20.07 h	20.07 h	20.11 h
96.	19.96 h	19.96 h	20.0 h
97.	19.09 h	19.09 h	19.13 h
98.	19.54 h	19.54 h	19.72 h
99.	19.2 h	19.2 h	19.24 h
100.	19.42 h	19.42 h	19.46 h

Table A.8: Comparison of the CFP-POI performance with and without POI scheduling using queries between northern Germany and central Italy and $\beta_s = 90$ % and $\beta_t = 10$ %.

ID	No POI scheduling	POI scheduling
1.	6.98 s	6.84 s
2.	8.4 s	6.98 s
3.	7.23 s	7.79 s
4.	7.83 s	7.2 s
5.	8.04 s	7.87 s
6.	7.23 s	6.74 s
7.	7.23 s	6.65 s
8.	7.84 s	7.01 s
9.	6.7 s	7.0 s
10.	10.12 s	8.06 s
11.	8.8 s	7.42 s
12.	6.89 s	5.8 s
13.	7.86 s	7.32 s
14.	7.98 s	7.17 s
15.	6.64 s	5.91 s
16.	9.5 s	7.09 s
17.	8.79 s	7.01 s
18.	7.12 s	6.7 s
19.	7.51 s	6.94 s
20.	8.19 s	7.54 s
21.	8.52 s	7.25 s
22.	9.76 s	7.08 s
23.	7.3 s	7.18 s
24.	6.23 s	6.89 s
25.	7.44 s	6.52 s
26.	7.32 s	7.29 s
27.	7.77 s	7.8 s
28.	7.54 s	6.79 s
29.	7.68 s	7.33 s
30.	7.21 s	7.35 s
31.	8.19 s	6.95 s
32.	7.36 s	6.09 s
33.	7.88 s	6.95 s
34.	8.18 s	7.01 s
35.	7.28 s	7.34 s
36.	8.28 s	6.18 s
37.	8.92 s	6.92 s
38.	7.76 s	8.33 s
39.	6.5 s	6.75 s
40.	6.9 s	5.92 s

Continued on next page

Table A.8 – continued from previous page

Query ID	No POI scheduling	POI scheduling
41.	7.97 s	7.06 s
42.	6.79 s	6.49 s
43.	7.16 s	6.53 s
44.	8.26 s	7.05 s
45.	7.81 s	6.94 s
46.	7.86 s	7.07 s
47.	8.04 s	6.44 s
48.	7.15 s	6.49 s
49.	9.54 s	9.11 s
50.	8.57 s	7.49 s
51.	7.37 s	7.95 s
52.	6.87 s	6.66 s
53.	7.5 s	8.07 s
54.	7.23 s	8.4 s
55.	8.89 s	7.55 s
56.	9.96 s	7.09 s
57.	7.59 s	8.57 s
58.	7.29 s	6.68 s
59.	7.37 s	6.41 s
60.	6.79 s	6.32 s
61.	8.17 s	6.63 s
62.	8.41 s	7.68 s
63.	7.76 s	6.75 s
64.	7.71 s	7.37 s
65.	6.92 s	7.47 s
66.	11.71 s	11.12 s
67.	6.78 s	6.93 s
68.	9.6 s	9.87 s
69.	9.34 s	14.41 s
70.	7.45 s	7.26 s
71.	8.06 s	6.7 s
72.	7.21 s	7.8 s
73.	12.99 s	13.68 s
74.	6.99 s	7.26 s
75.	7.29 s	6.78 s
76.	6.81 s	6.64 s
77.	7.66 s	7.3 s
78.	7.13 s	6.24 s
79.	7.18 s	6.83 s
80.	7.57 s	7.27 s
81.	7.02 s	6.61 s
82.	9.03 s	8.19 s

Continued on next page

Table A.8 – continued from previous page

Query ID	No POI scheduling	POI scheduling
83.	6.89 s	7.16 s
84.	7.38 s	7.84 s
85.	7.36 s	8.13 s
86.	8.03 s	7.91 s
87.	6.84 s	6.89 s
88.	7.26 s	7.0 s
89.	9.48 s	8.82 s
90.	7.41 s	6.73 s
91.	7.46 s	8.16 s
92.	6.54 s	6.56 s
93.	8.08 s	7.74 s
94.	7.99 s	6.93 s
95.	8.07 s	7.11 s
96.	10.26 s	9.62 s
97.	6.97 s	7.04 s
98.	7.5 s	6.89 s