

Electric Vehicle Charging Station Search in Stochastic Environments

Marianne Guillet^{1,2,3}, Gerhard Hiermann¹, Alexander Kröller², and Maximilian Schiffer¹

¹TUM School of Management, Technical University of Munich, 80333 Munich, Germany

² TomTom Location Technology Germany GmbH, 12435 Berlin, Germany

³ Chair of Operations Management, RWTH Aachen University, 52072 Aachen, Germany
marianne.guillet@tomtom.com, gerhard.hiermann@tum.de, alexander.kroeller@tomtom.com, schiffer@tum.de

Abstract

Electric vehicles are a central component of future mobility systems as they promise to reduce local noxious and fine dust emissions and CO₂ emissions, if fed by clean energy sources. However, the adoption of electric vehicles so far fell short of expectations despite significant governmental incentives. One reason for this slow adoption is the drivers' perceived range anxiety, especially for individually owned vehicles. Here, bad user-experiences, e.g., conventional cars blocking charging stations or inconsistent real-time availability data, manifest the drivers' range anxiety. Against this background, we study stochastic search algorithms, that can be readily deployed in today's navigation systems in order to minimize detours to reach an available charging station. We model such a search as a finite horizon Markov decision process and present a comprehensive framework that considers different problem variants, speed-up techniques, and three solution algorithms: an exact labeling algorithm, a heuristic labeling algorithm, and a rollout algorithm. Extensive numerical studies show that our algorithms significantly decrease the expected time to find a free charging station while increasing the solution quality robustness and the likelihood that a search is successful compared to myopic approaches.

Keywords: stochastic search, Markov decision process, dynamic programming, EV charging

1. Introduction

All around the world, governments and companies try to foster the adoption of electric vehicles (EVs), which are seen as a central component of future sustainable mobility systems that can play a major role in reducing noise, noxious, and carbon emissions, if fed by clean energy sources. While governments introduced national programs to support individual EV purchase, e.g., by means of ecological bonuses (France), tax exemption (Germany), or subsidies (Netherlands), private companies invested in public charging infrastructure (Bensasson 2019, Fröhlich 2019), and major logistic service providers progressively integrated EVs to realize sustainable transportation (DHL 2019, Amazon 2019). Yet the market uptake for EVs fell short in expectation in most countries. One of the main obstacles remaining, in particular for privately-owned EVs, is the customer’s perceived range-anxiety, triggered by missing charging infrastructure and incomplete or non-standardized information about charging services available. Hence, facilitating a reliable EV charging process is crucial to decrease range anxiety in order to invigorate the adoption of EVs (Bonges & Lusk 2016).

Different stakeholders focus on different measures to facilitate a reliable charging experience for users. At strategic level, municipalities try to facilitate reliable charging options by improving the overall charging infrastructure, e.g., by building new charging stations or increasing the capacity of existing ones. Further, enforcing new pricing schemes that support a higher turnover rate of (re)charged vehicles may help to save additional capacities (Bonges & Lusk 2016). However, infrastructure investments require long planning lead times and pricing schemes can hardly be realized without standardization and agreements between charging station operators. At operational level, some online map services exist and aim to help EV drivers to locate available charging stations. Unfortunately, such services struggle with data inaccuracy, e.g., an incomplete coverage with real-time status data. Even worse, charging stations can be blocked by non-charging vehicles (so-called “ICEing”) without the inaccessibility being reflected in status data. An empirical study in Berlin revealed a high correlation between such inaccessibility and parking availability, showing that in areas without available parking it is three times more likely for a charging station to be illegitimately blocked than legitimately used (see Table 1).

Drivers may find a selected and seemingly free station to be blocked when arriving there. Such negative user experiences may require additional detours with an already depleted battery in order to find a suitable charging station and thus may even increase a driver’s range anxiety. To this end, a reliable stochastic search algorithm that helps to route drivers to an available charging station as

Table 1.: Parking availability on data accuracy

Legal parking availability in vicinity	≥ 2 spots	1	0
Blocked by charging EV	5%	14%	10%
Blocked without connection (ICEing)	1%	9%	34%
Available	94%	77%	56%

Internal TomTom study in Berlin, 2019. The study shows how often charge points were legitimately used (i.e., “blocked” and actually used), actually available (correct “free”), or illegitimately blocked by a vehicle. The data relates to the amount of free parking spots found in the immediate vicinity. For most of the stations, real-time availability data was not available.

convenient as possible constitutes a valuable algorithmic component for today’s map and navigation services and may help to reduce the drivers’ range anxiety.

The goal of this paper is to develop such an algorithmic solution that can be readily deployed as a built-in implementation in today’s navigation systems and map services. In the remainder of this section, we first review the related literature, before we detail our contribution and elaborate the organization of this paper.

1.1. Literature review

We now survey literature on EV routing with uncertain charging stations before we focus on related search problems with stochastic resources.

Sweda et al. (2017) were the first to study probabilistic charging station availability as a multi-stop shortest path problem under uncertainty. They used a dynamic programming approach on a grid network and considered waiting times at unavailable stations. Kullman et al. (2017) extended this work by modeling an electric vehicle routing problem with uncertain charging at public charging stations, as a Markov decision process (MDP). Given the large action and state spaces, they proposed an approximate stochastic dynamic programming approach with a look-ahead procedure (Goodson et al. 2017). Jafari & Boyles (2017) studied a multicriteria stochastic shortest path problem for EVs, in which stochasticity relates to travel time and energy consumption.

Focusing on related stochastic search problems, Arndt et al. (2016) studied a probabilistic routing problem for on-street parking search with parking spots as stochastic resources and incorporated user preferences for stops closer to their destination. Besides showing the problem’s NP-completeness, they proposed a branch-and-bound algorithm for small problem spaces. More generally, Guo & Wolfson (2018) studied a probabilistic spatio-temporal resource search problem, in which resources have a general usage cost but the resource seeker is not allowed to wait for an occupied resource to become available again. Contrary to Arndt et al. (2016) in which resources observations are persistent during the search, stations can become available again after a defined time threshold. Guo & Wolfson (2018) proposed a value iteration solution procedure that remains tractable by making a fast recovery assumption at the instance level, which keeps the state space small. Schmoll & Schubert (2018) studied a dynamic resource routing problem under reliable real-time information, which requires fast (re)computations as resources frequently change their occupancy state. Assuming a large problem space, they used real-time dynamic programming to maintain utility values for likely states to determine the best action on-the-fly.

As can be seen, related research on stochastic charging station search is still scarce. Approaches that are specifically tailored to EV charging station search are limited to finding best cost paths between an origin and a destination rather than on an open search with an a-priori unknown destination. Further these approaches lack the consideration of sufficient real-world problem variants and heterogeneous station characteristics. Literature on stochastic resource search focus on an open search concept but lack EV and charging station specific characteristics. Moreover, the algorithmic solutions that have been proposed in this domain remain limited to a pure academic interest on artificial and small instances. Accordingly, these approaches cannot be embedded into real-time environments.

1.2. Contribution

With this work, we close the research gaps outlined above by providing a profound algorithmic framework that covers a wide range of problem variants in the context of stochastic EV charging station search, which can readily be deployed into real-time environments for online optimization purposes. Specifically, our contribution is fourfold. First, we formalize the problem of stochastic EV charging station search as a new stochastic search problem. We model this problem as a discrete finite horizon MDP and consider two objectives: minimizing i) the time until charging, and ii) the time until completion in case of heterogeneous charging times. Moreover, we present different problem variants in which a driver may wait or not at occupied stations and charging stations might be homogeneous or heterogeneous. Second, we prove the complexity of this problem class and show that it is NP-hard. Third, we develop an algorithmic tool chain that consists of three algorithms: an exact labeling algorithm for which we present a cost decomposition of the Markovian policy to derive effective dominance rules; a heuristic variant of this labeling algorithm which is amenable for real-time application; and a rollout algorithm. Additionally, we present multiple speed-up mechanisms, e.g., reduced action spaces and sharpened dominance relations. Fourth, we provide extensive numerical studies that base on real-world data for the cities of Berlin and San Francisco. Our results show that compared to myopic approaches, our search algorithms decrease the average time spent finding an available station by up to 44%. We further benchmark the performance of our heuristic algorithms against the exact algorithm and show that a combined, case-dependent usage of both allows for effective real-time application.

1.3. Organization

The remainder of the paper is as follows. In Section 2, we introduce the stochastic charge pole search (SCPS) problem. Section 3 formalizes our problem as an MDP and consecutively develops the corresponding algorithmic framework. In Section 4, we describe our case study and the experimental design. Section 5 discusses our numerical results. Section 6 concludes this paper and provides an outlook on future research. To keep this paper concise, we shift all proofs to Appendix A.

2. Problem definition and representation

In this section, we introduce the SCPS problem for which we specify four problem variants, each corresponding to a distinct real-world scenario.

2.1. Problem setting

We focus on a routing problem with stochastic charging station availability, where a driver starts at a given location and seeks to find an unoccupied charging station to recharge her vehicle. Her objective is to minimize her total expected cost, which consists of the driving time during the search and additional factors, e.g., the time until the charging is complete, or the time spent walking from the charge point to the actual destination. We consider this search to be spatially and temporally bounded to account for the driver’s limited time budget and penalize unsuccessful searches with a high termination cost to model the resulting discomfort.

Formally, we define this problem on a directed and complete graph $\mathcal{G} = (\mathcal{C}, \mathcal{A})$ that consists of a set of vertices \mathcal{C} and a set of arcs $(c, c') \in \mathcal{A}$. Each vertex $c \in \mathcal{C}$, except the vertex where the search starts, represents a charging station. The driver starts her search at a designated start vertex $c_0 \in \mathcal{C}$ at time 0, with a maximum search time of \bar{T} . We assume that the driver is willing to charge at any vertex in \mathcal{C} and consider a limited search radius by restricting \mathcal{C} to the stations within a maximum distance \bar{S} around the start vertex. Driving from c to c' takes $t_{c,c'} \geq 0$ units of time. We model the availability of a station $c \in \mathcal{C}$ at time $t \in [0, \bar{T}]$ as a visit-dependent binary random variable $a_c \in \{0, 1\}$, which the driver observes by visiting c . Without prior knowledge, i.e., when visiting a station for the first time, the probability that the station is free ($a_c = 1$) is a constant p_c . Table 2 summarizes this notation.

We introduce two time-based penalties: γ_c is the time-equivalent usage cost for using pole c if it is available upon arrival; β_c denotes the cost for unsuccessfully terminating the search at c in case c is occupied, which may happen if all vertices in \mathcal{C} have been unsuccessfully visited, or it is impossible to visit another candidate within the remaining time budget \bar{T} .

We define a solution to our problem as an ordered sequence of charging station visits $C = (c_0, \dots, c_n)$. In practice, a driver that uses this solution starts at c_0 and visits the charging stations in the given sequence up to the first available charging station. Accordingly, our search terminates either successfully at any $c \in C$ (with a total cost of driving time until that vertex, plus extra cost γ_c), or unsuccessfully at c_n (incurring driving time along C plus cost β_{c_n}). We denote by α the expected cost of solution C .

Problem setting variants: In practice, different system characteristics and user preferences may require to solve different problem variants, e.g., some drivers may want to find a charging station as fast as possible without considering station heterogeneity, while others may want to finish charging as quickly as possible. Such differences can be incorporated in our problem through the generic penalties β_c and γ_c , and an additional parameter \mathcal{W} , which states whether it is permitted to finish the search by waiting at a station $\mathcal{W} = \mathcal{W}^1$ or not $\mathcal{W} = \mathcal{W}^0$.

We identify four different problem variants that reflect the most common use cases in practical applications. Table 3 summarizes these variants and states the respective realizations as a triple $(\beta_c, \gamma_c, \mathcal{W})$. In variant $\neg\mathcal{W}/\neg C$, the driver is neither allowed to wait at stations nor do stations have (heterogeneous) usage costs. To this end, we consider a constant penalty cost $\bar{\beta}$ for an unsuccessful terminated search. While stations remain homogeneous, the driver is allowed to wait in variant $\mathcal{W}/\neg C$. We assume that an expected waiting duration W_c is known for every station c , and to be a constant independent of arrival time and earlier observations. In variants $\neg\mathcal{W}/C$ and \mathcal{W}/C , stations have

Table 2.: Notation used to define the SCPS

$\mathcal{G} = (\mathcal{C}, \mathcal{A})$	Network graph
\mathcal{W}	Permissible values for waiting decisions
L_c	Expected charging time at a charging station c
\bar{S}	Maximal distance allowed between any vertex and the origin vertex c_0
\bar{T}	Maximal overall driving time
W_c	Expected waiting time at an occupied charging station c
a_c	Binary random variable modelling c' availability
p_c	Initial probability that charging station c is available before any visit
$t_{c,c'}$	Driving time on arc (c, c')
β_c	Termination penalty cost at an occupied station c
γ_c	Termination cost at an available station c

Table 3.: Summary of problem setting variants and parameters $(\beta_c, \gamma_c, \mathcal{W})$

	No waiting allowed ($\neg W$)	Waiting allowed (W)
Charging time insensitive ($\neg L$)	$\neg W/\neg C = (\bar{\beta}, 0, \mathcal{W}^0)$	$W/\neg C = (W_c, 0, \mathcal{W}^1)$
Charging time sensitive (L)	$\neg W/C = (\bar{\beta}, L_c, \mathcal{W}^0)$	$W/C = (W_c + L_c, L_c, \mathcal{W}^1)$

heterogeneous charging durations and the driver seeks to minimize her total time to finish charging up to a preferred charge level. While the driver is not allowed to wait in $\neg W/C$ (i.e., the constant penalty cost $\bar{\beta}$ is induced in case of failure), this constraint is relaxed in W/C .

2.2. Discussion

While our model captures the essential characteristics of the underlying real-world problem, some comments are in order. First, we differentiate charging stations based on the charging duration in problem variants $\neg W/C$ and W/C . The model is however not restrictive and allows charging stations to be heterogeneous with respect to any other criteria, including walking distance to a destination location, price for charging, etc. Yet one needs to convert non time-based usage costs (e.g., charging prices) to time-based costs to ensure search cost compatibility. Second, we assume an occupied station to remain occupied during the search for most problem variants. This seems to be a plausible assumption based on the large difference between typical search times (minutes) and charging durations (hours) in an urban setting. However, we also present a problem variant with recovering probabilities in Section 3.4 and discuss its impact in Section 5. Third, our basic model is agnostic of the energy spent during the search. Here, a similar argument holds: given that a search typically covers only a small distance radius compared to a vehicle’s range, we assume this effect to be negligible. Nevertheless, we present a problem variant which considers the energy spent during the search. Our results for this problem variant show no significant impact of the respective energy consumption and thus confirm our assumption.

3. Methodology

We now introduce an MDP representation for the SCPS problem (Section 3.1) and a corresponding algorithmic framework. As the SCPS problem is NP-hard (see Appendix B) we develop one exact and two heuristic algorithms. We first present these algorithms for our basic problem variants and focus on an exact and a heuristic labeling algorithm in Section 3.2, before we present a rollout algorithm in Section 3.3. We then show how these algorithms must be modified to account for time dependent recovery probability functions (Section 3.4) and search-related energy consumption (Section 3.5).

3.1. Markov decision process

To model the SCPS problem as a finite-horizon MDP with multiple decisions over a time-budgeted process, we use the additional notation as summarized in Table 4.

Table 4.: Notation used to define the MDP

\mathcal{S}	State space
\mathcal{U}	Action space
x	State $\in \mathcal{S}$
$u(x)$	Action chosen in state $x \in \mathcal{S}$
w	Decision to wait (and charge) at the last visited station C if occupied
$d(u, u(x))$	Immediate cost induced by taking action u in state $x \in \mathcal{S}$
π	Policy
V^π	Value function, based on policy π
V^*	Optimal Value function
a	Binary variable modeling the availability of the last-visited station
δ	Binary variable indicating whether the selected station c in $x = (C, 0)$ is already contained in C
$t(\pi)$	Driving time following policy π
$\rho(\pi, k)$	Probability that at least one of the first k first stations of policy π is available
$\alpha(\pi)$	Total search cost for following policy π

3.1.1. Model variables and transition functions

We define a state x as a tuple (C, a) , where C is an ordered sequence of stations $C = (c_0, \dots, c_k)$ that have already been visited, with the driver being located at c_k . Let a be the binary realization of the availability of c_k , indicating whether c_k is free ($a=1$) or occupied ($a=0$). Then, the state space results to

$$\mathcal{S} = \{(C, a) : C = (c_0, \dots, c_k), c_j \in \mathcal{C} \forall j, a \in \{0, 1\}\}.$$

We denote by $u(x) = (c, w)$ a possible action taken in state $x \in \mathcal{S}$ to transition to state x' , with $w \in \mathcal{W}$ being a binary that indicates whether to wait at the current station ($w = 1$) or not ($w = 0$), and c stating the next station to visit. We note that, either $\mathcal{W} = \mathcal{W}^0 = \{0\}$ for problem variants without waiting or $\mathcal{W} = \mathcal{W}^1 = \{0, 1\}$ for problem variants that allow to wait at a charging station. Since we assume a station availability status to be persistent during the search, we further assume that for problem variants without waiting, each station should be visited at most once. For problem variants with waiting, given the heterogeneous termination costs β_c , we assume that the last station can be visited twice. Actions are only taken at occupied stations: if $a = 1$, x is a termination state and the search finishes as the driver charges at the found unoccupied station. Let $\tilde{\mathcal{C}}(C)$ be the restricted set of charging station vertices, reachable in less than $\bar{T} - \sum_{i=0}^{k-1} t_{c_i, c_{i+1}}$. Then, the action space for a state $x = (C, 0)$ results to

$$\mathcal{U}(x) = \{(c, w) : w \in \mathcal{W}, c \in \tilde{\mathcal{C}}, (\mathcal{W} = \mathcal{W}^0 \wedge \delta = 0) \vee (\mathcal{W} = \mathcal{W}^1 \wedge (\delta = 0 \vee (\delta = 1 \wedge w = 1)))\}, \quad (3.1)$$

with δ being the binary variable that indicates whether c is already included in C ($\delta = 1$) or not ($\delta = 0$). If $\tilde{\mathcal{C}}(C)$ is empty, no more station can be reached under the given time constraints and we refer to this state as a forced termination state.

We define the function $p_t(x' | x, u)$ to describe the probability that following action $u \in \mathcal{U}(x)$ from state $x \in \mathcal{S}$ would result in state $x' \in \mathcal{S}$, such that $\sum_{x' \in \mathcal{S}} p_t(x' | x, u) = 1$ and consider two cases:

1. If the selected action is to wait, then the driver stays at the current station. The transition is deterministic and $p_t((C, 1) | (C, 0), (c_k, 1)) = 1$ for a path C ending in c_k .
2. If the selected action is to continue searching, the selected next station c is either available or occupied with respect to $\tilde{p}_c(C)$. Hence $p_t((C', 1) | (C, 0), (c, 0)) = \tilde{p}_c(C)$ and $p_t((C', 0) | (C, 0), (c, 0)) = 1 - \tilde{p}_c(C)$, where C' is the sequence C extended by c .

We denote by $d(x, u)$ the immediate induced cost for taking action $u = (c, w)$ in state $x = (C, a)$, which depends on the realized availability a at the last station c_k and the respective action

$$d(x, u) = (1 - a)w\beta_{c_k} + a\gamma_{c_k} + (1 - w)(1 - a)t_{c_k, c}. \quad (3.2)$$

We note that if $a = 1$, the driver charges at station c_k , and $d(x, u)$ corresponds to the station usage cost γ_{c_k} . If $a = 0$ and the driver waits at the station, the search terminates and cost β_{c_k} results. If $a = 0$ and the search continues at the next station c , the cost results to the travel time $t_{c_k, c}$.

Finally, we define a policy π as a function mapping a state $x \in \mathcal{S}$ to an action $\pi(x) \in \mathcal{U}(x)$. Accordingly, π implicitly describes a search path $C(\pi) = (c_0, \dots, c_n)$ with $\pi(c_i, 0) = (c_{i+1}, 0) \forall i = 0, \dots, n-1$ and state $x_n = (C(\pi), a)$ that terminates the search at vertex c_n , either because the driver runs out of time or because she decides to wait with $\pi(x_n) = (c_n, 1)$. We refer to $C(\pi)$ as C and to $\tilde{p}_c(C)$ as \tilde{p}_c to keep the notation concise.

3.1.2. Cost function

We now analyze the cost function $V^\pi(C, a)$ that describes the expected cost for following a policy π , from a start state (C, a) . Then $V^\pi(x_0)$, with $x_0 = (c_0, 0)$ represents the expected search cost for the driver when following policy π from starting at vertex c_0 and the objective is to find a policy π that minimizes $V^\pi(x_0)$. Then, the cost function $V^\pi(C, a)$ can be expressed as follows

$$V^\pi(C, a) = a\gamma_{c_k} + (1 - a) [w\beta_{c_k} + (1 - w)(t_{c_k, c_{k+1}} + \tilde{p}_{c_{k+1}}V^\pi(C', 1) + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0))], \quad (3.3)$$

with $C = (c_0, \dots, c_k)$, $u(C, a) = (c_{k+1}, w)$, and $C' = C \cup \{c_{k+1}\}$. For both realizations of a , Equation 3.3 can be simplified:

$$V^\pi(C, 1) = \gamma_{c_k}, \quad (3.4)$$

$$V^\pi(C, 0) = w\beta_{c_k} + (1 - w) [t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0) + \tilde{p}_{c_{k+1}}\gamma_{c_{k+1}}]. \quad (3.5)$$

3.1.3. Cost structure variants

For each problem variant as introduced in Section 2.1, the cost functions $V^\pi(C, 0)$, $V^\pi(C, 1)$, and $V^\pi(x_k)$ for a termination state $x_n = (C, a)$ can be expressed as follows, with C' denoting the extension of C by station c_{k+1} for non-termination states.

No waiting, without charging ($\neg W/\neg C$):

$$\begin{aligned} V^\pi(C, 0) &= t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0), \\ V^\pi(C, 1) &= 0, \\ V^\pi(x_n) &= (1 - \tilde{p}_{c_n})\bar{\beta}. \end{aligned} \quad (3.6)$$

Waiting permitted, without charging ($W/\neg C$):

$$\begin{aligned} V^\pi(C, 0) &= wW_{c_k} + (1 - w)[t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0)], \\ V^\pi(C, 1) &= 0, \\ V^\pi(x_n) &= (1 - \tilde{p}_{c_n})W_{c_n}. \end{aligned} \quad (3.7)$$

No waiting, with charging ($-W/C$):

$$\begin{aligned} V^\pi(C, 0) &= t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0) + \tilde{p}_{c_{k+1}}L_{c_{k+1}}, \\ V^\pi(C, 1) &= L_{c_k}, \\ V^\pi(x_n) &= (1 - \tilde{p}_{c_n})\bar{\beta} + \tilde{p}_{c_n}L_{c_n}. \end{aligned} \quad (3.8)$$

Waiting permitted, with charging (W/C):

$$\begin{aligned} V^\pi(C, 0) &= w(L_{c_k} + W_{c_k}) + (1 - w)[t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', 0) + \tilde{p}_{c_{k+1}}L_{c_{k+1}}], \\ V^\pi(C, 1) &= L_{c_k}, \\ V^\pi(x_n) &= (1 - \tilde{p}_{c_n})W_{c_n} + L_{c_n}. \end{aligned} \quad (3.9)$$

3.1.4. Cost function expansion

We now expand V^π to derive an explicit evaluation of the search cost. To simplify notation, we use $C_{[i:j]}$ to denote a sub-sequence (c_i, \dots, c_j) of a sequence $C = (c_0, \dots, c_i, \dots, c_j, \dots, c_n)$.

Proposition 1. *Let $C = (c_0, \dots, c_n)$. Then, the cost for being in state $x_k = (C_{[0:k]}, 0)$, with $k < n$, following policy π until the termination state $x_n = (C, a)$ expands as follows*

$$\begin{aligned} V^\pi(x_k) &= \prod_{i=k}^n (1 - \tilde{p}_{c_i}(C_{[k:i-1]}))\beta_{c_n} + \sum_{i=k}^{n-1} \left[t_{c_i, c_{i+1}} \prod_{j=k}^i (1 - \tilde{p}_{c_j}(C_{[k:j-1]})) \right] \\ &+ \sum_{i=k}^n \left[\gamma_{c_i} \tilde{p}_{c_i}(C_{[k:i-1]}) \prod_{j=k}^{i-1} (1 - \tilde{p}_{c_j}(C_{[k:i-1]})) \right]. \end{aligned} \quad (3.10)$$

Given that policies encode solutions, we can express the solution cost α for a policy, and set $\alpha(\pi) = V^\pi(x_0)$. Following Equation 3.10 and Proposition 1, this yields

$$\begin{aligned} \alpha(\pi) &= \prod_{i=0}^n (1 - \tilde{p}_{c_i}(C_{[0:i-1]}))\beta_{c_n} + \sum_{i=0}^{n-1} \left[t_{c_i, c_{i+1}} \prod_{j=0}^i (1 - \tilde{p}_{c_j}(C_{[0:j-1]})) \right] \\ &+ \sum_{i=0}^n \left[\gamma_{c_i} \tilde{p}_{c_i}(C_{[0:i-1]}) \prod_{j=0}^{i-1} (1 - \tilde{p}_{c_j}(C_{[0:j-1]})) \right]. \end{aligned} \quad (3.11)$$

For a more concise notation, let $\bar{\rho}(\pi, k)$ be the probability that a driver fails in finding at least one free station in $C_{[0:k]}$, while $\rho(\pi, k) = 1 - \bar{\rho}(\pi, k)$ is the probability that she succeeds in finding at least one free station in $C_{[0:k]}$

$$\bar{\rho}(\pi, k) = \prod_{i=0}^k (1 - \tilde{p}_{c_i}(C_{[0:i-1]})), \quad (3.12)$$

with $\tilde{p}_{c_i}(C_{[0:i-1]})$ denoting the likelihood that c_i is available after having visited all previous stations from c_0 to c_{i-1} . Furthermore, let

$$A(\pi) = \sum_{i=0}^{n-1} [t_{c_i, c_{i+1}} \bar{\rho}(\pi, i)] + \sum_{i=0}^n \gamma_{c_i} \tilde{p}_{c_i}(C_{[0:i-1]}) \bar{\rho}(\pi, i-1). \quad (3.13)$$

Then, we rewrite Equation 3.11 as

$$\alpha(\pi) = \bar{\rho}(\pi, n)\beta_{c_n} + A(\pi). \quad (3.14)$$

We denote by $t(\pi) = \sum_{i=0}^{n-1} t_{c_i, c_{i+1}}$ the accumulated driving time for all stations in C and note that for a feasible solution, $t(\pi) \leq \bar{T}$ holds.

3.2. Dynamic programming based labeling algorithms

To find an optimal solution for the SCPS problem, we develop a dynamic programming based labeling algorithm. Similar to solving multi-criteria constrained shortest path problems, we propagate partial policies in order to find an in-expectation cost-optimal policy. Herein, we use a dominance criterion to withdraw non-promising partial policies early to keep the explored search space as small as possible. Formally, we associate each partial policy π_c with a label L_c whose resources depend on the problem variant. For no-waiting variants $\neg W/\neg C$ and $\neg W/C$, a label $L_c = (t_c, A_c, \rho_c, \alpha_c, S_c)$ consists of the accumulated driving time t_c , the partial cost A_c (cf. Equation 3.14), the likelihood ρ_c to successfully finish the search up to vertex c (cf. Equation 3.12), the total cost α_c , and the set of reachable and non-visited poles S_c . For waiting variants $W/\neg C$ and W/C , we add an additional resource R_c that denotes the set of reachable but visited poles.

To describe our labeling algorithm, we denote by \mathcal{L}^a the set of active labels and by L_0 the initial label that corresponds to our start location. Let $\mathcal{F}_{cc'}(L)$ be a set of resource extension functions (REFs) that expand a label L whose partial policy ends at vertex c to a label L' whose partial policy ends at vertex c' . Let $\alpha(L)$ be the cost associated with label L . We define $\delta^+(L)$ as a function that returns a set of tuples (c, c') which denotes all feasible physical successor locations $c \in \mathcal{C}$ for a label L whose partial policy ends at $c \in \mathcal{C}$.

Using this notation, Figure 1 shows a pseudo code of our dynamic programming algorithm. We initialize our list of active labels \mathcal{L}^a and our so far best found solution L^* with L_0 (1.1) and start propagating labels until our search terminates when \mathcal{L}^a is empty (1.2). We then process labels in \mathcal{L}^a in cost increasing order (1.3). Once a label got selected for propagation, we remove it from \mathcal{L}^a (1.4) and propagate it considering all of its feasible successors (1.5) using the REFs (1.6). We check whether a newly created label L' is dominated by an existing label in \mathcal{L}^a (1.7). If this is not the case, we remove labels which are dominated by L' from \mathcal{L}^a (1.8) and add L' to \mathcal{L}^a respectively (1.9). Whenever the newly created label is a termination label, i.e., its corresponding state indicates that the driver should wait or there are no feasible successors left, we check if the found label improves the so far best found label and update L^* accordingly (1.10&11).

In the remainder of this section, we detail the REFs used to extend a label and the dominance criterion to discard a dominated label.

Resource extension functions: To extend a label L which corresponds to a partial policy ending at vertex $c \in \mathcal{C}$ to a new label L' which corresponds to a partial policy ending at vertex $c' \in \mathcal{C}$, we write

Figure 1.: Dynamic programming based labeling algorithm.

```

1:  $\mathcal{L}^a \leftarrow \{L_0\}$ ,  $L^* \leftarrow L_0$ 
2: while  $\mathcal{L}^a \neq \emptyset$  do
3:    $L \leftarrow \text{costMinimumLabel}(\mathcal{L}^a)$ 
4:    $\mathcal{L}^a \leftarrow \mathcal{L}^a \setminus \{L\}$ 
5:   for  $(c, c') \in \delta^+(L)$  do
6:      $L' \leftarrow \mathcal{F}_{ij}(L)$ 
7:     if  $\text{isNotDominated}(L', \mathcal{L}^a)$  then
8:        $\text{dominanceCheck}(\mathcal{L}^a, L')$ 
9:        $\mathcal{L}^a \leftarrow \mathcal{L}^a \cup \{L'\}$ 
10:      if  $((w' == 1) \vee (\delta^+(L') = \emptyset)) \wedge \alpha(L') < \alpha(L^*)$  then
11:         $L^* \leftarrow L'$ 
12: return  $L^*$ 

```

$L' \leftarrow \mathcal{F}_{c'}(L)$ and use the following set $\mathcal{F}_{c'}$ of REFs :

$$A_{c'} = A_c + (1 - \rho_c)(t_{c,c'} + \tilde{p}_{c'}\gamma_{c'}) \quad (3.15)$$

$$1 - \rho_{c'} = (1 - \rho_c)(1 - \tilde{p}_{c'}) \quad (3.16)$$

$$t_{c'} = t_c + t_{c,c'} \quad (3.17)$$

$$S_{c'} = S_c \setminus \{c'\} - \bigcup_{d \in S_c, t_{c',d} > \bar{T}} \{d\} \quad (3.18)$$

$$R_{c'} = R_c \cup \{c'\} - \bigcup_{d \in R_c, t_{c',d} > \bar{T}} \{d\} \quad (3.19)$$

$$\alpha_{c'} = A_{c'} + (1 - \rho_{c'})\beta_{c'} \quad (3.20)$$

Equation 3.15 propagates partial cost A_c along arc (c, c') with respect to its definition (cf. Equation 3.13) considering the arc-dependent driving time $t_{c,c'}$, the availability probability $\tilde{p}_{c'}$ and usage cost $\gamma_{c'}$ of vertex c' . Equation 3.16 propagates the success rate ρ_c (cf. Equation 3.12) based on the availability probability of vertex c' . The accumulated driving time $t_{c'}$ is straightforwardly propagated along arc (c, c') considering the arc-dependent driving time $t_{c,c'}$ (Equation 3.17). To obtain the set $S_{c'}$ from S_c (Equation 3.18), we remove from S_c the last visited vertex c' and all vertices $d \in S_c$ that cannot be reached from vertex c' within the remaining search time $\bar{T} - t_{c'}$, i.e., such that $t_{c',d} \geq \bar{T} - t_{c'}$. To obtain $R_{c'}$ (Equation 3.19), we insert the visited vertex c' in R_c and similarly subtract all vertices $d \in R_c$ that cannot be reached from vertex c' within the remaining search time. We propagate cost straightforwardly by Equation 3.20.

Dominance criterion: Equation 3.14 decomposes the non-monotonous cost $\alpha(\pi)$ into monotonous resources $A(\pi)$ and $\rho(\pi)$ that partly define a label. Given the monotonicity of $A(\pi)$, $\rho(\pi)$, and $t(\pi)$, we can then consider two partial policies π_1 and π_2 that end with the same vertex visit c and their associated labels L_1 and L_2 and say that for problem variants $\neg W/\neg C$ and $\neg W/C$, L_1 dominates L_2 ($L_1 \succ L_2$), if the following conditions are true:

$$1 - \rho_c(\pi_1) \leq 1 - \rho_c(\pi_2) \quad (3.21)$$

$$A_c(\pi_1) \leq A_c(\pi_2) \quad (3.22)$$

$$t_c(\pi_1) \leq t_c(\pi_2) \quad (3.23)$$

$$\forall c' \in S_c(\pi_2), c' \in S_c(\pi_1) \quad (3.24)$$

$$\forall c' \in S_c(\pi_1), \bar{T} - t_c(\pi_1) + p_{c'}(\gamma_{c'} - \bar{\beta}) \leq 0 \quad (3.25)$$

Here, conditions (3.21)–(3.23)&(3.25) ensure that the cost and duration of π_1 is smaller than the cost of π_2 . Conditions (3.23)&(3.24) check whether all non-visited vertices reachable by π_2 can be reached from π_1 as well. Condition (3.25) checks whether all reachable stations from c for π_1 contribute to decrease $\alpha_c(\pi_1)$. This check is necessary for settings where π_1 can be extended with k more stations than π_2 to detect corner cases in which these k additional stations may increase $\alpha(\pi_1)$ to a value larger than $\alpha(\pi_2)$.

For problem variants $W/\neg C$ and W/C , we recall that a search can terminate before the whole time budget is spent and that a station terminating the search might have been visited earlier. Accordingly we drop condition 3.25 and introduce condition 3.26 that checks whether all visited vertices reachable by π_2 can be reached from π_1 as well. Recalling that the label reads $L_c = (t_c, A_c, \rho_c, \alpha_c, S_c, R_c)$, we

then say that for problem variants $W/-C$ and W/C , $L_1 \succ L_2$ if conditions (3.21)–(3.24) are true and

$$\forall c' \in R_c(\pi_2), c' \in R_c(\pi_1) \cup S_c(\pi_1) \quad (3.26)$$

holds. While Algorithm 1 solves the SCPS problem optimally with the dominance criterion above, one may consider to drop some of the dominance conditions to obtain a heuristic dominance criterion that withdraws more labels at the price of losing optimality. In the remainder of this paper, we study a heuristic labeling algorithm, where we preserve conditions (3.21)&(3.22) to obtain cost dominance but neglect conditions (3.23),(3.24),(3.25) and (3.26). In this context, the label definition simplifies to $L_c = (A_c, \rho_c, \alpha_c)$. We provide evidences to the selection of this heuristic dominance criterion in Section 5.2.

3.3. Rollout algorithm

In this section, we introduce a rollout algorithm, built as a forward dynamic programming procedure. We identify the best action at a given state as the one that yields minimal approximated cost. Here, the core idea of the cost approximation is to greedily expand the current policy from each candidate action until a defined horizon to obtain an associated cost value via backpropagation. We first detail the main procedure for $-W/-C$ and $-W/C$ variants before we show how to account for additional waiting decisions in problem variants $W/-C$ and W/C . We apply a one-step decision rollout strategy (cf. Goodson et al. 2017) whose complexity equals a post-state decision rule as the approximation reduces to $w = 0$ decisions.

Let k be the index of the k^{th} decision epoch and let x_k be the non terminated epoch's state with $x_k = (C = (c_0, c_1, \dots, c_k), 0)$ and c_k being the last station visited in epoch k . Let $x_{k+1} = (C', 0)$ be the state in the $(k+1)^{\text{th}}$ epoch that results from action $u = (c, 0)$ at epoch k , with $u \in \mathcal{U}(x_k)$. Using this additional notation, Figure 2 details the pseudo-code of our algorithm.

We initialize the sequence of station visits C with the start vertex (1.2) and expand the sequence until the time horizon is reached (1.3). From the current state x_k , we seek to determine the next best action $(c^*, 0)$ and initialize the variables that encode it (1.4). For all possible succeeding states x_{k+1} (1.5), we use a heuristic policy $\tilde{\pi}_c$ that bases on a greedy procedure to propagate state x_{k+1} up to a forced terminated state x_{k+K} , with a look-ahead of K epoch extensions. For all propagated states x_l with $l \in [k+1, K-1]$, the greedy procedure chooses the action $(c_{l+1}, 0)$, with station c_{l+1} being selected

Figure 2.: Forward programming based algorithm.

```

1:  $c_k \leftarrow c_0, C \leftarrow (c_0), x_k \leftarrow (C, 0), t \leftarrow 0$ 
2: while  $t \leq \bar{T}$  do
3:    $c^* \leftarrow 0, x^* \leftarrow 0, C^* \leftarrow 0, Q \leftarrow \infty$ 
4:   for  $(c, 0) \in \mathcal{U}(x)$  do
5:      $x_{k+1} \leftarrow (C', 0)$ 
6:      $V \leftarrow \text{greedyCost}(x_{k+1})$ 
7:      $Q(x_k, c, x_{k+1}) \leftarrow t_{c_k, c} + (1 - \tilde{p}_c)V + \tilde{p}_c \gamma c$ 
8:     if  $Q(x_k, c, x_{k+1}) < Q$  then
9:        $Q \leftarrow Q(x_k, c, x_{k+1})$ 
10:       $c^* \leftarrow c, x^* \leftarrow x, C^* \leftarrow C'$ 
11:     $C \leftarrow C^*, x_k \leftarrow x^*, t \leftarrow t + t_{c_k, c^*}, c_k \leftarrow c^*$ 
12: if  $w' == 1$  then
13:    $C \leftarrow \text{refinePolicy}(C)$ 
14: return  $C$ 

```

based on availability probabilities of vertices reachable from c_l and the driving time to each of these vertices. We then use these anticipated states to evaluate the expected value of the policy-specific cost $V^{\tilde{\pi}_c}(x_{k+1})$ for the candidate state x_{k+1} . We define $\mathbf{GreedyCost}(x_{k+1})$ as the function that carries out the greedy expansion from $x_{k+1} = (C', 0)$ and returns $V^{\tilde{\pi}_c}(x_{k+1})$ (1.6&1.7). Repeating the greedy procedure and cost evaluation for all possible next actions $u = (c, 0) \in \mathcal{U}(x_k)$ allows us to find the action $u = (c, 0)$ that minimizes the cost to transition from state x_k to state x_{k+1} (1.8) that we define as $Q(x_k, c, x_{k+1})$ (cf. Equation 3.29). Eventually, the selected action $u = (c^*, 0)$ yields minimal cost $Q(x_k, c^*, x_{k+1})$ (1.10-1.12). Only for variants $W/-C$ and W/C , $\mathbf{refinePolicy}(C)$ bases on the resulting visits sequence C to introduce the waiting decision at the best stage $1 \leq k \leq n$, with n being the total amount of station visits in C (1.13).

In this setting, the reduced action space for problem variants $-W/-C$ and $-W/C$ reads

$$\tilde{\mathcal{U}}(x_k) = \{(c, w) : c \in \mathcal{C}, w = 0\} \quad (3.27)$$

and we calculate the cost for being in state x_{k+1} , $V^{\tilde{\pi}_c}(x_{k+1})$, based on the greedy policy $\tilde{\pi}_c$ as

$$V^{\tilde{\pi}_c}(x_{k+1}) = \prod_{l=k+1}^K (1 - \tilde{p}_{c_l}) \beta_{c_l} + \sum_{l=k+1}^{K-1} [d(x_l, \tilde{\pi}_c(x_l))] \prod_{m=k+1}^l (1 - \tilde{p}_{c_m}), \quad (3.28)$$

which allows to derive $Q(x_k, c, x_{k+1})$ as

$$Q(x_k, c, x_{k+1}) = t_{c_k, c} + (1 - \tilde{p}_c) V^{\tilde{\pi}_c}(x_{k+1}) + \tilde{p}_c \gamma_c. \quad (3.29)$$

For problem variants $W/-C$ & W/C , we first calculate the no-waiting case and compute policy π with the associated ordered sequence of charging stations $C = (c_0, \dots, c_n)$. We denote π as an intermediate policy and introduce π_S representing the final search policy. Then, $\mathbf{refinePolicy}(C)$ calculates π_S using the intermediate policy π while permitting wait-actions. For each intermediary charging station c_k at the k^{th} decision epoch, $c_k \in (c_0, \dots, c_n)$, $k \neq n$, π provides a sub sequence of charging stations to visit until the end of the search (c_{k+1}, \dots, c_n) and thus the policy-specific cost value $V^\pi(x_k)$, associated to policy π and state $x_k = ((c_0, \dots, c_k), 0)$. We aim to quantify for each station c_k whether the termination cost β_{c_k} is actually lower than the expected cost for continuing the search $V^\pi(x_k)$. If this is the case, the optimal decision is to wait and we refine π into π_S with $\pi_S(x_k) = (c_k, w = 1)$ and $V^{\pi_S}(x_l) \leq V^\pi(x_l) \forall l \in [0, n]$.

We define π_S as

$$\pi_S(x_k) = \arg \min_{\pi(x_k), (c_k, w=1)} w \beta_{c_k} + (1 - w) [t_{c_k, c_{k+1}} + (1 - \tilde{p}_{c_{k+1}}) \min(V^\pi(x_{k+1}), V^{\pi_S}(x_{k+1})) + \tilde{p}_{c_{k+1}} \gamma_{c_{k+1}}], \quad (3.30)$$

where $\pi(x_k) = (c_{k+1}, w = 0)$. If there exists an index k such that $0 \leq k < n$, $x_k = ((c_0, \dots, c_k), 0)$ and $\pi_S(x_k) = (c_k, 1)$, then state x_k terminates the search, as the driver will wait at c_k if c_k is not immediately available. In this case π_S encodes the solution (c_0, \dots, c_k) .

3.4. Time-dependent probability recovery function

In the basic setting of the SCPS problem, we assume that a station does not change its availability during the search's time horizon and restrict the amount of visits to each individual station. We now relax these assumptions and show which modifications are necessary to consider a recovery function r_c that allows the availability of an occupied station c to recover over time. In this case, we define $\tilde{p}_c(C)$

as follows

$$\tilde{p}_c(C) = \begin{cases} p_c, & \text{if } c \notin C \\ r_c(\Delta_c) & \text{otherwise,} \end{cases} \quad (3.31)$$

where $\Delta_c = \sum_{j=l}^{k-1} t_{c_j, c_{j+1}} + t_{c_k, c}$, with l denoting the position of the last visit to c in C .

We still consider a charging station c to be initially available at a probability of p_c . When it is blocked at the driver's arrival time $t_c = \sum_{j=0}^{\ell-1} t_{c_j, c_{j+1}}$, it may become available over time, i.e., the availability probability of c recovers according to $r_c(\Delta_c)$, which denotes a station's availability probability for an arbitrary point in time $t = \Delta_c + t_c$ that remains after the first visit and before the end of the time horizon. We specify $r_c(\Delta)$ for any Δ based on Schmoll & Schubert (2018) as

$$r_c(\Delta) = \frac{\mu_c}{\lambda_c + \mu_c} (1 - e^{-(\mu_c + \lambda_c)(\Delta)}). \quad (3.32)$$

Here, $\frac{1}{\lambda_c}$ and $\frac{1}{\mu_c}$ denote the average time station c remains available, respectively occupied and remain constant over the search's time horizon. We can then express p_c as a function of λ_c and μ_c , with $p_c = \frac{\mu_c}{\lambda_c + \mu_c}$ (cf. Jossé et al. 2015) and simplify Equation 3.32 to

$$r_c(\Delta_c) = p_c (1 - e^{-(\frac{\mu_c}{p_c})(\Delta_c)}). \quad (3.33)$$

We now assume that stations can be visited as many times as needed. Then, the action space for a state $x = (C, 0)$ is slightly modified as follows,

$$\mathcal{U}(x) = \{(c, w) : c \in \tilde{\mathcal{C}}(C), w \in \mathcal{W}\}.$$

In the remainder of this section, we outline the changes that are necessary to adapt the labeling algorithm to such a setting. Apart from the new availability probability definition $\tilde{p}_c(C)$, no modifications are necessary for the rollout algorithm.

Modifications for the Labeling Algorithm: The MDP definition remains unchanged, because we can determine the arrival time at a station based on the arrival time at the preceding station extended by the driving time that remains deterministic. Thus Δ_c and $\tilde{p}_c(C)$ can be calculated from C without any further information. However, we need to modify the initial dominance criterion (3.21)–(3.25)(resp. (3.21)–(3.26)) as the optimal solution may now contain multiple visits to any charging station, independent on the problem variant.

We consider two partial policies π_1 and π_2 that end with the same visit at vertex c and their associated labels L_1 and L_2 . We recall that $R_c(\pi_1)$ denotes the set of charging stations which have been already visited by π_1 . We then say $L_1 \succ L_2$, if (3.21)–(3.26) hold and

$$\frac{1 - \rho_c(\pi_1)}{1 - \rho_c(\pi_2)} \leq \prod_{c' \in R_c(\pi_1)} \frac{(1 - p_{c'}(1 - e^{-(\mu+\lambda)(\Delta_{c'}(\pi_2))}))}{(1 - p_{c'}(1 - e^{-(\mu+\lambda)(\Delta_{c'}(\pi_1))}))} \quad (3.34)$$

holds. Here, Equation 3.34 accounts for different probabilities of charging stations that have already been visited in both policies at different points in time by considering the biggest possible difference of probability values for c' between both paths. We note that we leave the heuristic variant of the dominance criterion unchanged in this context.

3.5. Integrating search related energy consumption

In this setting, we assume that the vehicle starts its search with an initial state of charge (SoC) b_0 , which reduces over the course of the search depending on the driven distances. Then, longer driving distances

result in higher energy consumption and an additional trade-off results between visits to far-distanced stations with a high availability probability and visits to near-distanced stations with medium to low availability probabilities. Here, a higher availability probability may be counterbalanced by traveling longer distances, which increases the energy that must be recharged or limits future visits to potential stations accordingly.

To account for this setting, we introduce additional notation and denote the energy consumed when traversing arc (c, c') as $k_{c,c'}$ and a vehicle's SoC after having visited all stations in $C = (c_0, \dots, c_k)$ as b . In this setting, we can only transition from state $(C, b, 0)$ using arc (c_k, c_{k+1}) if $b \geq k_{c_k, c_{k+1}} + \underline{b}$, with \underline{b} denoting the minimum feasible state of charge a vehicle must keep. Analogously to common monotonicity assumptions in related settings (cf. Sweda et al. 2017), we assume that

$$t_c(\pi_1) < t_c(\pi_2) \iff b_c(\pi_1) \geq b_c(\pi_2)$$

and state the necessary MDP modifications which hold as follows

$$\mathcal{S} = \{(C, b, a) : C = (c_0, \dots, c_k), c_k \in \mathcal{C} \forall k, b \in [0, q_{max}], a \in \{0, 1\}\}, \quad (3.35)$$

$$\begin{aligned} \mathcal{U}(x) = \{(c, w) : w \in \mathcal{W}, c \in \bar{\mathcal{C}}, \\ (\mathcal{W} = \mathcal{W}^0 \wedge \delta = 0) \vee (\mathcal{W} = \mathcal{W}^1 \wedge (\delta = 0 \vee (\delta = 1 \wedge w = 1)))\}, \end{aligned} \quad (3.36)$$

$$\begin{aligned} V^\pi((C, b, a)) = (1 - a)w\beta_{c_k} + a\gamma_{c_k} + (1 - w)(1 - a)[t_{c_k, c_{k+1}} \\ + (1 - \tilde{p}_{c_{k+1}})V^\pi(C', b - k_{c_k, c_{k+1}}, 0) + \tilde{p}_{c_{k+1}}V^\pi(C', b - k_{c_k, c_{k+1}}, 1)]. \end{aligned} \quad (3.37)$$

First, we include a vehicle's SoC into the state space (Equation 3.35). Second, we modify the action space such that it depends on $\bar{\mathcal{C}}(C)$, which denotes a restricted set of charging station vertices that are reachable from state (C, a) in less than $\bar{T} - t(\pi)$ time, with $k_{c, c'} + \underline{b} \leq b$ and $t(\pi) = \sum_{k=0}^n t_{c_k, c_{k+1}}$ (Equation 3.36). The policy-specific cost function results straightforwardly from the modified action and state spaces (Equation 3.37). To account for this new setting in our algorithms, the following modifications are necessary.

Variants $\neg W/\neg C$ and $W/\neg C$: The rollout algorithm requires no modification. For the labeling algorithms, sets S_c and R_c now denote vertices that are reachable within the remaining time budget and within the remaining energy budget b_c . Accordingly, we add to the existing dominance conditions (3.21)–(3.25) or (3.21)–(3.26) that

$$b_c(\pi_1) \geq b_c(\pi_2), \quad (3.38)$$

must hold. The heuristic dominance relation remains unchanged, since the modified setting does not affect conditions (3.21)&(3.22).

Variants $\neg W/C$ and W/C : Here, the charging duration L_c at station c now depends on the amount of time needed to recharge from b_0 up to the maximum SoC \underline{b} . We introduce δL_c to denote the additional charging time due to the battery depletion δb_c during the search, where $\delta b_c = b_0 - b_c = \sum_{l=0}^{l=i} k_{c_l, c_{l+1}}$. We then account in each algorithm for the charging duration as

$$L'_c = L_c + \delta b_c \frac{L_c \bar{b}}{(1 - b_0)}. \quad (3.39)$$

3.6. Computational complexity improvements

In this section, we proof additional characteristics that allow to improve the computational complexity of certain problem variants. We first introduce three action space reductions, before we focus on a sharpened dominance relation for both the exact and the heuristic labeling algorithm.

3.6.1. Action space reductions

In the following, we discuss some action space reductions, which we summarize in Table 5. We refer to the initially defined action space as *complete*. In addition, we create the following reduced action spaces.

direct: To further reduce the search space, we restrict the visits from the last visited station c_k to feasible neighbor stations c such that there does not exist any station c' on the shortest path from c_k to c and denote $\tilde{\mathcal{A}}(C)$ the set of all feasible arcs (c_k, c) . We however allow c to be visited multiple times and show with Proposition 2 that accordingly *direct* doesn't lead to a loss of generality for problem variants $\neg W/\neg C$ and $W/\neg C$.

direct/restricted: For very large instances of the problem, we combine the *direct* action space with the visits restrictions from *complete*. The setting significantly reduces the search space but at the expense of optimality for problem variants $\neg W/\neg C$ and $W/\neg C$.

T^r -restricted: Finally, we restrict visits from station c_k to stations c reachable in less than T^r time units, i.e., $t_{c_k,c} \leq T^r$ while preserving the visit restriction of *complete*.

Proposition 2. *Action space \mathcal{U} can be modified such that $\mathcal{U}(C, 0) = \{(c) : (c_k, c) \in \tilde{\mathcal{A}}\}$ without loss of generality for $\gamma_c = 0 \forall c \in \mathcal{C}$.*

3.6.2. Sharpened dominance relation for the dynamic programming algorithms

In the following, we aim to sharpen the dominance relation, i.e., we aim to discard labels faster without dropping optimality. In our initial problem setting, we assume that both travel times and charging station availability probabilities are unbounded. We now account for a bound on each of these values that still reflects a real-world application. Specifically, we assume that *i*) the EV driver must travel at least a certain amount of time between two charging stations

$$0 < \underline{t} \leq t_{c,c'}, \forall c, c' \in \mathcal{C} \times \mathcal{C},$$

and *ii*) that one can never be entirely sure that a charging station is available

$$p_c \leq \bar{p} < 1, \forall c \in \mathcal{C}.$$

Table 5.: Modified action spaces

<i>complete</i>	$\mathcal{U}((C, 0) = \{(c, w) : w \in \mathcal{W}, c \in \tilde{\mathcal{C}}, (\mathcal{W} = \mathcal{W}^0 \wedge \delta = 0) \vee (\mathcal{W} = \mathcal{W}^1 \wedge (\delta = 0 \vee (\delta = 1 \wedge w = 1)))\})$
<i>direct</i>	$\mathcal{U}((C, 0) = \{(c, w) : (c_k, c) \in \tilde{\mathcal{A}}(C), w \in \mathcal{W}\})$
<i>direct/restricted</i>	$\mathcal{U}((C, 0) = \{(c, w) : (c_k, c) \in \tilde{\mathcal{A}}(C), \delta = 0, w \in \mathcal{W}\})$
<i>T^r-restricted</i>	$\mathcal{U}((C, 0) = \{(c, w) : w \in \mathcal{W}, c \in \tilde{\mathcal{C}}, (\mathcal{W} = \mathcal{W}^0 \wedge \delta = 0) \vee (\mathcal{W} = \mathcal{W}^1 \wedge (\delta = 0 \vee (\delta = 1 \wedge w = 1)))\}, t_{c_k,c} \leq T^r\})$

These bounds can be computed during a preprocessing step and allow for a sharpened dominance relation for the dynamic programming based labeling algorithms without invalidating our generic model. We note that charging and waiting times at a station are bounded as well. Let W_{min} and L_{min} (respectively W_{max} and L_{max}) be the minimal (respectively maximal) waiting and charging times.

We then consider two partial policies π_1 and π_2 that end with the same vertex visit and their associated labels L_1 and L_2 and we refine the initial dominance criterion (3.21)–(3.25) (resp. (3.21)–(3.26)) into stronger dominance checks. Let $\Delta A_c = A_c(\pi_1) - A_c(\pi_2)$, $\Delta \bar{\rho}_c = \bar{\rho}_c(\pi_1) - \bar{\rho}_c(\pi_2)$ and $\Delta \alpha_c = \alpha_c(\pi_1) - \alpha_c(\pi_2)$

In the initial setting, if $\Delta \bar{\rho}_c \leq 0$ and $\Delta A_c > 0$, i.e., Equation (3.21) holds but Equation (3.22) not, we cannot conclude that L_1 dominates L_2 . In the new setting, we can ensure that if quantity $\Delta \alpha_c$ is small enough while quantity $\bar{\rho}_c$ is large enough, then $L_1 \succ L_2$, as the lower bounds on $t_{c,c'}$ and upper bounds on p_c bound the propagated values of α_c and ρ_c . We then say $L_1 \succ L_2$, if (3.21)&(3.23)–(3.25)(resp. (3.26)) still hold and

$$\bar{p} \Delta \alpha_c \leq (-\Delta \bar{\rho}_c)(\underline{t} + \bar{p}(\gamma_{min} - \beta_{max})) . \quad (3.40)$$

Table 6 summarizes all variant-specific parameters β_{max} , γ_{min} .

Table 6.: Parameter affectation for each problem variant

Parameter	$\neg W/\neg C$	$W/\neg C$	$\neg W/C$	W/C
β_{max}	β	W_{max}	β	$W_{max} + L_{max}$
γ_{min}	0	0	L_{min}	L_{min}

4. Design of experiments

To benchmark our algorithms, we develop real-world instances that allow for extensive simulation experiments. We consider three different spatial patterns (see Figure 3), based on the west side of San Francisco, USA (*SF-1*), the city center of Berlin, Germany (*BER-1*), and the financial district of San Francisco (*SF-2*). Here, we account for free-flow speeds to calculate travel times $t_{c,c'}$ that denote the time-shortest path between two stations c and c' . As our search algorithms appear to be rather insensitive to the search’s starting point, we randomly choose one starting point for each pattern and use this starting point in every instance that builds on the respective pattern.

Besides the significant sensitivity to the instance size given by \bar{S} and \bar{T} , we found during preliminary studies that our search algorithms are sensitive to two general instance characteristics: the search area’s charging station density and the charging station availability probability. Accordingly, we use the patterns described above to create a set of instances that covers a broad parameter range for these characteristics. Figure 4 shows the amount of charging stations for each pattern, depending on the search radius \bar{S} around each patterns starting point.

Our TomTom internal availability study in the city of Berlin (cf. Table 1) shows on average high charging stations availability in areas with a large number of on-street parking spots. In areas with few available parking spots, drivers often use available stations as free parking spots, thus blocking access to EV drivers. In the study, the sole parking availability factor largely impacts the station availability. To reflect these amplitudes, we introduce three availability settings, drawing probabilities p_c for each charging station c from a β -distribution, which is centered on an expected availability of



Each subplot shows the geographic area used to build the respective instance graph. *SF-1* represents the west side of San Francisco, USA, *BER-1* the city center of Berlin, Germany, and *SF-2* the financial district of San Francisco. This figure bases on Open Charge Map contributors (2019) data, which is data licensed under CC BY-SA 4.0¹.

Figure 3.: City maps

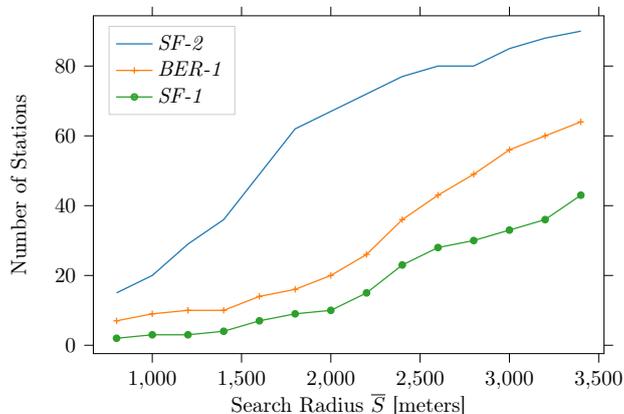


Figure 4.: Amount of stations per spatial scenario

[0.15, 0.60, 0.90] to consider a low- (*low-15%*), medium- (*avg-60%*) and high-availability (*high-90%*) scenario. The *avg-60%* and *high-90%* settings represent areas with average to high parking availability. The *low-15%* setting depicts a fictitious extreme case scenario prospectively corresponding to stricter parking policies and allows us to evaluate our algorithm behavior in such an environment.

For problem variants $W/-C$ and W/C , we consider a waiting duration W_c for each station c , which is uniformly distributed in [3, 15, 60, 120] minutes. For problem variants $-W/C$ and W/C , we consider a charging duration L_c , uniformly distributed in [30, 60, 120] minutes, to account for heterogeneous stations.

With this setup, we create a total of 9 scenarios by combining each area (*BER-1*, *SF-1*, *SF-2*) with each availability probability distributions (*low-15%*, *avg-60%*, *high-90%*). We then consider a small and a large-size instance for each combination, defined by the search's time budget and search radius (\bar{T} [min]/ \bar{S} [meters]). For denser areas, *BER-1* and *SF-2*, this corresponds to (5/800), resp. (10/2000). For the sparse area *SF-1*, since an 800 meters-area only comprises 2 stations we consider slightly wider search spaces (10/2000), respectively (15/2600) for *SF-1*, which derives a set of 18 instances per variant, 72 in total. We denote for each area, the smaller-size instance by *area/1* (e.g., *SF-1/1*) and the larger-size instance by *area/2*.

¹For more information see <https://creativecommons.org/licenses/by-sa/4.0/>

For our studies, we set the penalty cost to $\bar{\beta} = 120$ minutes for $\neg W/\neg C$ and to $\bar{\beta} = 200$ minutes for $\neg W/C$ and refer to Section 5.2 for a discussion on its selection and sensitivity. To evaluate our algorithms, we conduct $N = 1000$ simulation runs, each with a different station availability realization, drawn from the respective probability distribution. For all average values reported in our results discussion, we applied two-tailed Wilcoxon signed-rank tests to verify its significance.

5. Results

In the following, we discuss our results. We first detail the performance of our algorithms with respect to several quality metrics as well as their computational effort (Section 5.1), before we discuss results of additional sensitivity analysis and modeling assumptions (Section 5.2). We implemented the proposed algorithms single-threaded with Python 3.6.9, using PyPy 7.3.0 with GCC 7.3.1 and performed all experiments on a Virtual Machine on a Hypervisor, with 19 cores and 60 GB of RAM, running Ubuntu 18.04.3 LTS. For the following discussion we note that all algorithms can be implemented more efficiently, e.g., in a C++ environment. However, as this work stems from an industry project, we are not allowed to disclose the C++ related computational times for confidentiality reasons and use the Python implementation for a fair discussion of the algorithms' complexity.

5.1. Performance analysis

We discuss the algorithms' performance from two perspectives. First, we analyze the practical benefit of our algorithms and compare their solution quality based on their performance in our simulation environment. We then relate these results to a technically focused discussion on their computational complexity.

During algorithm testing and development, we noticed that both the labeling heuristic and the rollout algorithm are sensitive to the used action space (cf. Section 3.6) in between different problem variants. For the following discussion, we report results for the respective best-performing reduced action space and refer to Appendix D for a detailed discussion of this impact.

Applicability

In practice, more than one key performance indicator exists to evaluate the performance of our algorithms. In the following we refer to $\hat{\alpha}$ as the realized search cost, which corresponds to the realized driving time needed to find a station for variants $\neg W/\neg C$ and $W/\neg C$ and to complete charging for $\neg W/C$ and W/C . For W variants, $\hat{\alpha}$ includes the waiting time whenever the last visited station is occupied. Specifically, we analyze for each algorithm its

average search cost deviation which we calculate as

$$\Delta\hat{\alpha} = \sum_{i=1}^{1.000} \frac{(\hat{\alpha}_i - \hat{\alpha}_i^*)/\hat{\alpha}_i^*}{1.000}$$

with $\hat{\alpha}_i$ being the realized search cost of the respective algorithm for simulation run c and $\hat{\alpha}_i^*$ being the best realized search cost for simulation run c out of all algorithms;

success rate that results straightforwardly from the number of simulations, for which the respective search strategy finally led to a free charging station;

maximum search cost deviation that results to $\Delta\hat{\alpha}^{\max} = \hat{\alpha}^{\max}/\hat{\alpha}^{\max*}$ with $\hat{\alpha}^{\max} = \max \hat{\alpha}_i$ being the maximum cost out of all simulation runs for a single algorithm and with $\hat{\alpha}^{\max*}$ being the maximum cost over all simulation runs and all algorithms.

As the exact labeling algorithm (LE) cannot solve large instances of our problem to optimality, we limit the comparison to our heuristic algorithms (LH and RO) and add a naive and a greedy solution as myopic benchmarks. The greedy search (G) creates a sequence of charging station visits based on a greedy cost combining the travel time weighted by a station’s availability probability and a penalty cost weighted by its occupancy probability. The naive search (N) mimics a driver without assistance by selecting the closest non-visited station with highest availability probability.

Tables 7–9 detail the average search cost deviations between all algorithms (Table 7), the success rate of each algorithm (Table 8), and the maximum search cost deviation between all algorithms (Table 9), based on 1.000 simulation runs for each problem variant and scenario to avoid a statistical bias.

Table 7 shows that the RO and the LH algorithm significantly outperform the naive and the greedy benchmark in terms of average search costs. Here, an average deviation of zero implies that an algorithm always found the best search strategy, while increasing deviations indicate that the best search strategy was not found by the respective algorithm. For scenarios with high and average charging station availability, RO and LH show a similar performance. For scenarios with a low charging station availability, we observe higher differences that result in significant deviations for specific problem

Table 7.: Average search cost deviations

		<i>low-15%</i>				<i>avg-60%</i>				<i>high-90%</i>			
		N	G	RO	LH	N	G	RO	LH	N	G	RO	LH
<i>-W/-C</i>	<i>SF-1/1</i>	1.38	0.78	0.00	0.77	2.63	0.41	0.00	0.00	1.33	0.35	0.19	0.00
	<i>SF-1/2</i>	1.13	0.59	0.09	0.14	2.66	2.55	0.00	0.00	2.52	1.08	0.00	0.00
	<i>BER-1/1</i>	0.08	0.02	0.02	0.08	0.39	0.39	0.25	0.27	0.28	0.98	0.00	0.00
	<i>BER-1/2</i>	0.18	0.08	0.01	0.01	3.38	0.16	0.03	0.03	4.14	0.24	0.00	0.00
	<i>SF-2/1</i>	4.74	4.72	0.57	0.28	5.98	5.98	0.01	0.01	6.51	3.18	0.01	0.01
	<i>SF-2/2</i>	8.31	6.52	0.25	0.16	10.7	6.16	0.17	0.10	6.43	6.43	0.01	0.01
<i>W/-C</i>	<i>SF-1/1</i>	5.67	6.61	6.68	0.60	2.71	1.72	0.01	0.01	1.08	0.12	0.00	0.00
	<i>SF-1/2</i>	2.30	6.19	0.67	0.41	2.71	1.69	0.01	0.01	2.53	0.11	0.00	0.00
	<i>BER-1/1</i>	0.41	1.32	0.20	0.20	0.15	0.36	0.04	0.04	0.25	0.00	0.00	0.00
	<i>BER-1/2</i>	4.39	1.35	0.20	0.20	4.36	0.62	0.24	0.24	4.12	0.00	0.00	0.00
	<i>SF-2/1</i>	58.1	4.55	0.42	0.42	6.41	0.55	0.01	0.01	6.52	0.04	0.01	0.01
	<i>SF-2/2</i>	33.1	4.72	0.19	0.19	10.8	0.73	0.17	0.09	6.40	0.05	0.01	0.05
<i>-W/C</i>	<i>SF-1/1</i>	0.12	0.02	0.05	0.03	1.10	0.01	0.00	0.00	0.98	0.07	0.00	0.00
	<i>SF-1/2</i>	1.41	0.18	0.11	0.20	2.85	0.02	0.01	0.00	0.06	0.05	0.00	0.00
	<i>BER-1/1</i>	0.20	0.19	0.07	0.04	0.13	0.01	0.01	0.01	2.76	0.00	0.00	0.00
	<i>BER-1/2</i>	0.42	0.35	0.05	0.21	0.94	0.03	0.01	0.01	2.92	0.06	0.00	0.00
	<i>SF-2/1</i>	1.24	0.01	0.12	0.09	0.21	0.03	0.00	0.00	1.02	0.02	0.00	0.00
	<i>SF-2/2</i>	0.19	0.06	0.03	0.02	0.99	0.03	0.00	0.00	1.03	0.03	0.00	0.03
<i>W/C</i>	<i>SF-1/1</i>	2.37	1.17	0.00	0.00	1.14	0.04	0.00	0.00	0.98	0.00	0.00	0.00
	<i>SF-1/2</i>	2.11	4.24	0.01	0.01	2.83	0.04	0.00	0.00	0.06	0.00	0.00	0.00
	<i>BER-1/1</i>	0.74	0.01	0.00	0.00	0.12	0.02	0.00	0.00	2.79	0.02	0.00	0.00
	<i>BER-1/2</i>	1.37	0.01	0.01	0.01	0.94	0.02	0.00	0.00	2.92	0.00	0.00	0.00
	<i>SF-2/1</i>	3.48	0.03	0.01	0.01	0.17	0.01	0.00	0.00	1.02	0.00	0.00	0.00
	<i>SF-2/2</i>	0.89	0.04	0.02	0.02	0.98	0.01	0.00	0.00	1.02	0.00	0.00	0.00

The table compares the average search cost deviation $\Delta\hat{\alpha}$ for LH, RO, G and N for each of the 72 instances.

Table 8.: Success rate

		<i>low-15%</i>				<i>avg-60%</i>				<i>high-90%</i>			
		N	G	RO	LH	N	G	RO	LH	N	G	RO	LH
$\neg W/\neg C$	<i>SF-1/1</i>	0.81	0.85	0.36	0.86	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-1/2</i>	0.95	0.85	0.81	0.85	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>BER-1/1</i>	0.78	0.77	0.77	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>BER-1/2</i>	0.95	0.80	0.78	0.80	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-2/1</i>	0.70	0.80	0.78	0.86	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-2/2</i>	0.90	0.86	0.87	0.88	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
$\neg W/C$	<i>SF-1/1</i>	0.80	0.55	0.79	0.83	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-1/2</i>	0.94	0.74	0.69	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>BER-1/1</i>	0.78	0.73	0.78	0.78	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00
	<i>BER-1/2</i>	0.93	0.92	0.68	0.93	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-2/1</i>	0.74	0.51	0.42	0.77	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	<i>SF-2/2</i>	0.89	0.89	0.91	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

The table compares the success rate $\hat{\rho}$ for LH, RO, G and N for each instance of $\neg W$ problem variants.

variants and classes. As can be seen, higher search cost deviations occur for the W variants, especially for $W/\neg C$, due to the heterogeneous penalty costs that result from waiting at an occupied station. Particularly, on the *SF-1* instances at low-availability, RO performs significantly worse compared to LH with respect to the average search time. In this case, a single varying station visit between two search strategies can cause such differences due to the limited amount of candidate stations and the large amplitude between penalty costs.

Table 8 shows the success rate of each algorithm for each problem variant and scenario. We disclose only $\neg W$ problem variants in this table, because W problem variants always end successfully; in the worst case with a long waiting time at the last visited charging station. As can be seen, the LH algorithm shows much higher success rates compared to the RO algorithm, which highlights the superiority of the LH algorithm compared to the RO algorithm as both algorithms show comparable performances with respect to average search cost. Remarkably, for some scenarios, the naive approach yields the highest success rates. This highlights that there exists a trade off between search costs and success rates for $\neg W$ problem variants. Analyzing Tables 7&8 jointly, we conclude that for some scenarios the naive algorithm outperforms the LH algorithm in terms of success rates at the price of significantly higher search cost. Vice versa, the RO algorithm outperforms the LH algorithm for some scenarios in terms of average search costs at the price of significantly reduced success rates (see, e.g., $\neg W/\neg C$, scenario *SF-1/1*).

Table 9 shows the deviation between the maximum search cost of all algorithms for W problem variants, which are guaranteed to be feasible at the price of high waiting cost. Here, we observe a trade-off between the average search cost and the maximum search cost. As one can see, the LH and the RO algorithm show a similar performance for scenarios with medium and high charging station availability, but the LH algorithm performs better in the *SF-1/1* scenario with low charging station availability. Across all scenarios the greedy algorithm sometimes yields the least maximum cost. However, this improvement stems from significantly increased search costs (cf. Table 7).

Figure 5 illustrates the trade-offs discussed above by showing for all algorithms the trade off between average search costs and i) the success rate, aggregated over all scenarios for $\neg W$ problem variants

(Figure 5a), or ii) the maximum search cost, aggregated over all scenarios for W problem variants (Figure 5b).

Table 9.: Maximal Search cost

		<i>low-15%</i>				<i>avg-60%</i>				<i>high-90%</i>			
		N	G	RO	LH	N	G	RO	LH	N	G	RO	LH
$W/\neg C$	$SF-1/1$	22.6	2.29	10.8	0.00	1.42	0.00	0.10	0.24	2.11	0.00	0.13	0.13
	$SF-1/2$	4.45	2.29	0.00	0.00	2.01	0.11	0.00	0.00	0.69	0.00	0.13	0.13
	$BER-1/1$	1.12	0.00	0.32	0.32	0.28	2.08	0.00	0.00	0.27	0.34	0.00	0.00
	$BER-1/2$	33.7	0.00	0.32	0.32	0.60	2.08	0.00	0.00	0.71	0.34	0.00	0.00
	$SF-2/1$	29.0	0.27	0.00	0.00	0.89	1.05	0.00	0.00	0.74	0.00	0.70	0.70
	$SF-2/2$	30.2	0.27	0.00	0.00	3.60	2.04	0.00	1.07	0.02	0.55	0.00	0.55
W/C	$SF-1/1$	6.07	1.22	0.00	0.00	2.60	0.00	0.01	0.01	0.85	0.00	0.00	0.00
	$SF-1/2$	2.78	6.24	0.00	0.00	2.61	0.01	0.00	0.00	0.00	0.03	0.03	0.03
	$BER-1/1$	2.69	0.00	0.00	0.00	0.73	0.36	0.00	0.00	1.63	1.70	0.00	0.00
	$BER-1/2$	6.41	0.00	0.00	0.00	0.96	1.38	0.00	0.00	2.39	0.01	0.00	0.00
	$SF-2/1$	6.34	0.06	0.00	0.00	2.57	0.00	0.00	0.00	0.99	0.00	0.01	0.01
	$SF-2/2$	6.49	0.20	0.00	0.00	0.85	0.00	0.02	0.01	0.95	0.00	0.01	0.00

The table compares the maximal search costs $\Delta\hat{\alpha}^{\max}$ for LH, RO, G and N for each instance of W problem variants.

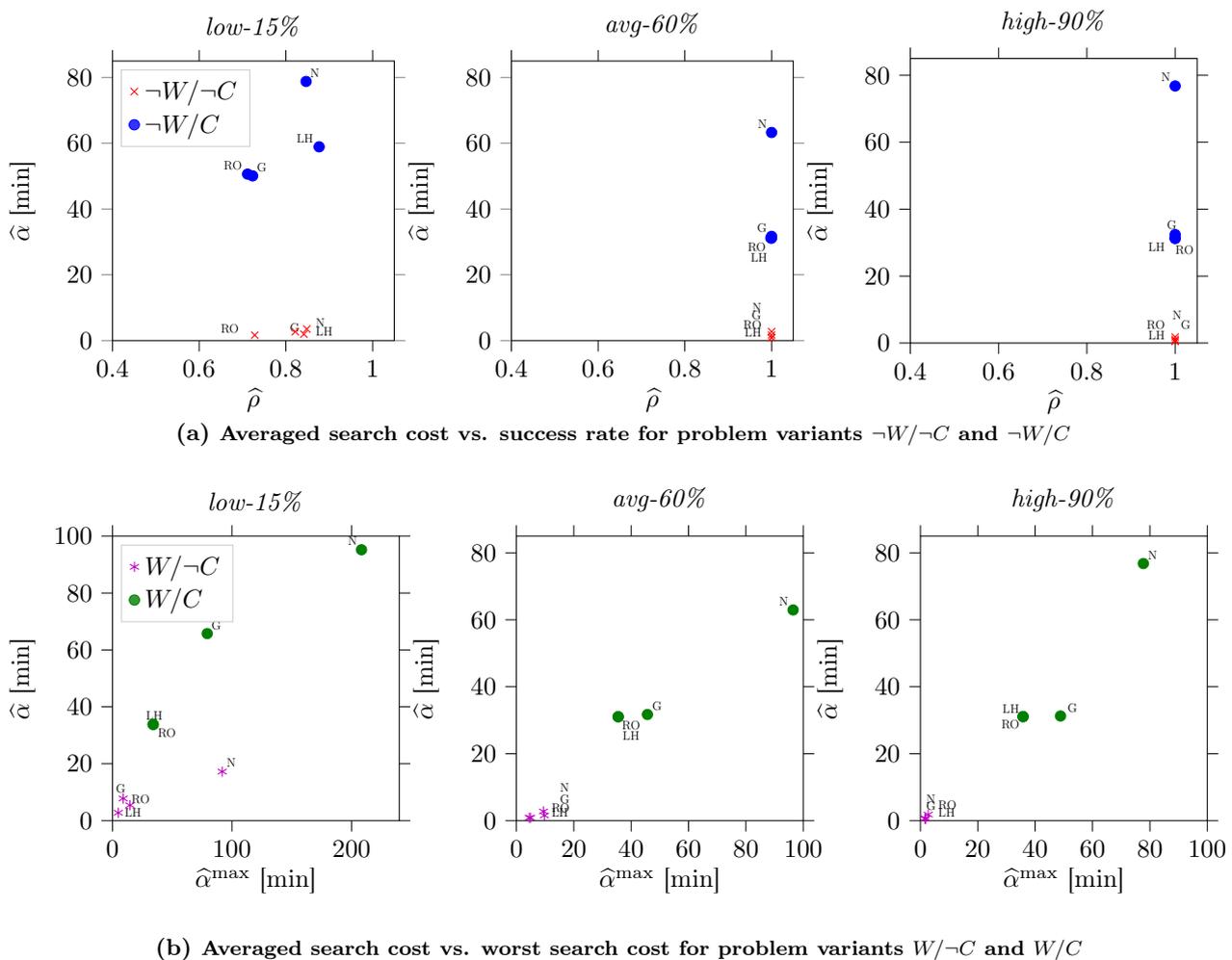


Figure 5.: Trade off between the average search cost and the success rate, resp. the worst search cost

For $\neg W$ problem variants and scenarios with medium or high charging station availability, all algorithms yield a 1.00% success rate but the advanced algorithms yield significantly lower cost, with LH being the best-performing algorithm. In these cases the greedy algorithm performs close to the advanced algorithms for problem variant $\neg W/C$. For scenarios with low charging station availability the LH algorithm yields a better success rate but the RO algorithm yields lower search cost. For W problem variants, we observe that the advanced algorithms outperform the myopic algorithms significantly, independent of the charging station availability. This performance difference is higher for problem variant $W/\neg C$. For problem variant W/C , G performs close to the advanced algorithms for scenarios with medium or high charging station availability.

Concluding, our results show that the developed search algorithms can significantly improve the search quality across all problem variants and scenarios. Compared to G, the advanced algorithms decrease the search cost by 21 % in average and up to 44% for areas with a scarce number of charging stations and low charging station availability. For $\neg W$ variants, the failure rate decreases by 30 % with low charging station availability. Moreover, advanced algorithms allow to reduce search times by 5 ($W/\neg C$) to 31 (W/C) minutes compared to myopic approaches for W problem variants with low station availability. This reduction potential decreases for scenarios with average to high station availability. However, advanced algorithms appear to be more robust in these cases and lower the worst search costs by 20 % ($W/\neg C$) and 12 % (W/C). Comparing the LH and RO algorithm among each other, LH prolongs the search compared to RO but obtains a significantly higher success rate for $\neg W$ variants. For scenarios *avg-60%* and *high-90%* RO and LH show a similar performance.

Computational tractability

To compare the performance of the LH and RO algorithm against the exact labeling algorithm, we derive a set of 8064 test instances by varying $\bar{T} \in [5, 10, 15, 20]$ minutes, $\bar{S} \in [800, 1.000, \dots, 3400]$ meters, and the availability distribution in $\{low-15\%, avg-60\%, SF-2\}$ for all problem variants and each search area. Here, we use a large time limit of 15,000 seconds to obtain a maximum set of solutions that are eligible for our comparison, i.e., solutions for instances that could be solved with all three algorithms.

Table 10 shows the rate of instances solved within this time limit (\hat{n}), and the average computational time of the successful runs (\hat{t}) for both heuristic algorithms (RO, LH) and for the exact labeling algorithm (LE), as well as the average ratio (g_α) between each heuristic result and the optimal solution.

As can be seen, our LH algorithm provides optimal or close to optimal solutions. The RO algorithm shows a similar solution quality for problem variants of type W but shows a significantly worse solution quality for problem variants of type $\neg W$ with a low charging station availability. While the RO algorithm succeeds in solving all instances with computational times of a few seconds, the LH algorithm improves upon its exact counterpart with respect to the number of instances solved and computational times but can neither solve all instances nor preserve computational times at the order of seconds.

Figure 6 illustrates the solution quality deviations between RO and LH for a representative subset of scenarios (*BER-1*, $\bar{T} = 10$) with low and medium charging station availability and varying search radii $\bar{S} \in [800, 1.000, \dots, 3400]$. As can be seen, the observed differences are sensitive to the problem variant. Significant differences occur for $\neg W$ problem variants with low charging station availability and varying search radii, while the algorithms perform similarly on all other problem variants. These high deviations result from penalty costs for unsuccessful searches, which can only result for $\neg W$ problem variants and are more likely to occur at low charging station availability.

Table 10.: Aggregated computational results over all tested instances per scenario

		LH			RO			L-E		LH			RO			L-E	
		g_α	\hat{t}	\hat{n}	g_α	\hat{t}	\hat{n}	\hat{t}	\hat{n}	g_α	\hat{t}	\hat{n}	g_α	\hat{t}	\hat{n}	\hat{t}	\hat{n}
		$-W/-C$							$-W/C$								
<i>low-15%</i>	<i>SF-1</i>	1.01	1.78	1.00	1.32	0.09	1.00	165	0.64	1.00	104	1.00	1.10	0.04	1.00	87.7	0.64
	<i>BER-1</i>	1.04	0.49	1.00	1.43	0.28	1.00	1386	0.46	1.00	412	1.00	1.43	0.18	1.00	1338	0.46
	<i>SF-2</i>	1.03	445	0.66	1.22	1.38	1.00	8646	0.04	1.06	1088	0.70	1.48	0.86	1.00	5313	0.02
<i>avg-60%</i>	<i>SF-1</i>	1.00	266	0.96	1.04	0.11	1.00	7.02	0.64	1.00	9.34	1.00	1.05	0.05	1.00	7.12	0.64
	<i>BER-1</i>	1.00	84.5	1.00	1.01	0.23	1.00	473	0.59	1.00	0.54	1.00	1.03	0.15	1.00	1057	0.64
	<i>SF-2</i>	1.00	803	0.34	1.11	1.76	1.00	4816	0.05	1.00	355	0.57	1.00	0.83	1.00	5560	0.25
<i>high-90%</i>	<i>SF-1</i>	1.00	106	0.66	1.03	0.10	1.00	16.4	0.64	1.05	75.1	0.93	1.01	0.05	1.00	8.48	0.64
	<i>BER-1</i>	1.00	66.7	0.66	1.00	0.35	1.00	517	0.57	1.00	2.82	1.00	1.00	0.15	1.00	266	0.59
	<i>SF-2</i>	1.00	1.70	0.25	1.00	0.95	1.00	4156	0.04	1.00	750	0.34	1.00	1.06	1.00	2303	0.04
		$W/-C$							W/C								
<i>low-15%</i>	<i>SF-1</i>	1.04	0.05	1.00	1.09	0.01	1.00	482	0.68	1.00	1.90	1.00	1.00	0.12	1.00	493	0.68
	<i>BER-1</i>	1.00	1.60	1.00	1.00	0.04	1.00	283	0.64	1.00	5.98	1.00	1.00	0.77	1.00	803	0.66
	<i>SF-2</i>	1.00	160	0.64	1.03	0.10	1.00	2111	0.25	1.00	358	0.79	1.00	2.95	1.00	2565	0.25
<i>avg-60%</i>	<i>SF-1</i>	1.00	0.05	1.00	1.03	0.01	1.00	2.33	0.64	1.00	0.25	1.00	1.00	0.12	1.00	473	0.68
	<i>BER-1</i>	1.00	0.72	1.00	1.00	0.01	1.00	1046	0.64	1.00	0.83	1.00	1.00	0.57	1.00	495	0.64
	<i>SF-2</i>	1.00	490	0.54	1.09	0.03	1.00	4219	0.25	1.00	527	0.64	1.00	2.59	1.00	2031	0.25
<i>high-90%</i>	<i>SF-1</i>	1.00	0.06	1.00	1.00	0.01	1.00	1.68	0.64	1.00	1.99	0.89	1.00	0.08	1.00	352	0.68
	<i>BER-1</i>	1.00	0.38	1.00	1.00	0.01	1.00	476	0.64	1.00	0.05	0.89	1.00	0.34	1.00	630	0.66
	<i>SF-2</i>	1.00	0.43	0.25	1.00	0.05	1.00	2024	0.25	1.00	4.27	0.29	1.00	1.88	1.00	2661	0.25

Abbreviations hold as follows: g_α - averaged optimality gap over all tested instances with $g_\alpha = \frac{\alpha_{\text{heur}}}{\alpha_{\text{opt}}}$, \hat{t} - averaged computational time[s], \hat{n} - rate of instances that can be computed in less than 15000 seconds. We note that an average g_α of 1.00 indicates that an algorithm (almost) always finds the optimal solution. If it always finds the optimal solution we highlight the respective g_α in bold font, whereas we leave it in normal font if some solutions remain heuristic but are not reflected in the value of g_α due to rounding.

Figure 7 shows the computational time of LH, RO, and LE for a representative subset of instances (*BER-1*, $\bar{T} = 20$, *high-90%*) for different search radii. As can be seen, RO and LH remain equally efficient for search radii up to 1400m but the computational time of LH increases exponentially for bigger search radii. Synthesizing Table 10 and Figures 6&7, we observe a trade-off between LH and RO: while RO yields robust computational times at the price of varying solution quality, LH yields a robust solution quality at the price of exponentially increasing computational time.

From a practitioner’s perspective, computational times of a few milliseconds are imperative to deploy a search algorithm in practice, e.g., embedded into a navigation application. Here, one could resolve the trade-off between RO and LH in two different ways. On the one hand, one could apply both algorithms selectively, using LH for tractable problem sizes and RO for larger problem sizes. On the other hand, one could always apply LH and terminate its search after a given time limit. To analyze which strategy appears to be more promising in our case, we compare the performance of RO and LH against each other, limiting the computation time of LH to 1 second, which equals a sufficiently small computational time when using an efficient implementation.

Figure 8 shows this comparison for problem variant $-W/-C$, while we provide figures for all other problem variants in Appendix E to keep this paper concise. In general, the time-limited LH outperforms

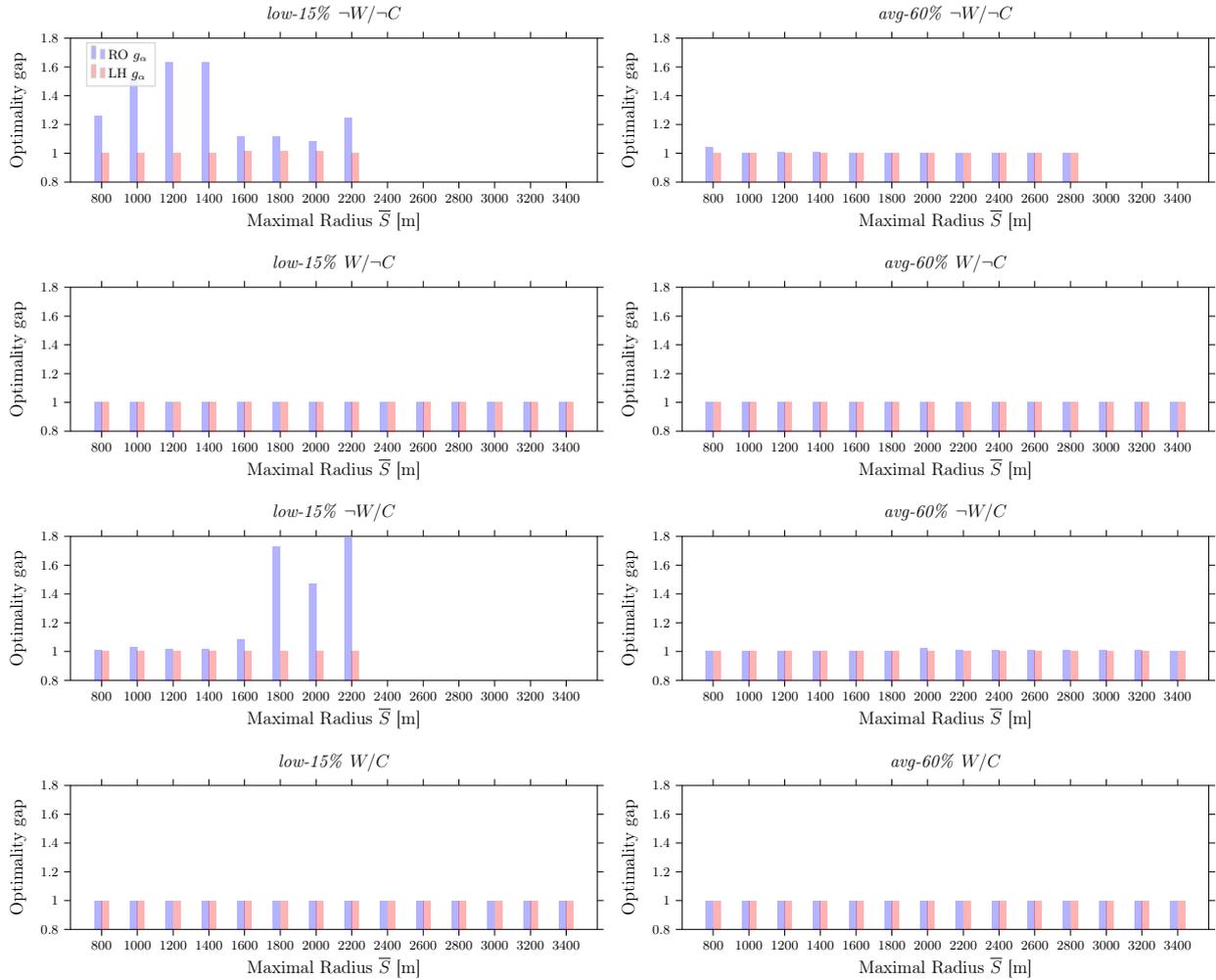


Figure 6.: Optimality gap for LH and RO for the *BER-1* scenario with fixed $\bar{T}=10$ min for 15 % and 60 % availability scenarios

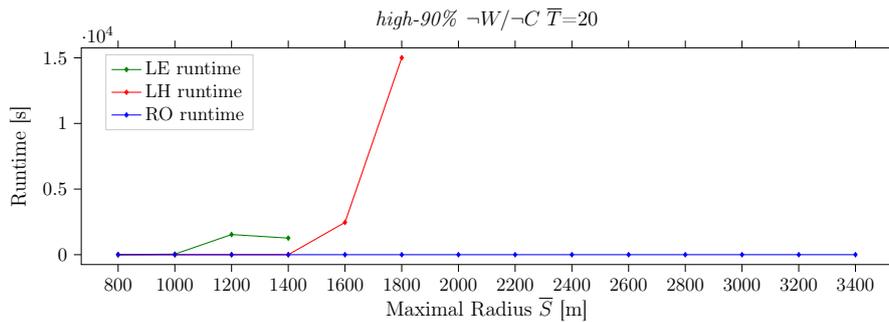


Figure 7.: Computational times for LH, LE and RO for the *BER-1* scenario with fixed $\bar{T}=20$ min with 90 % availability

RO (blue areas) for small search radii, in particular for instances with low charging station availability, whereas RO outperforms the time-limited LH (red areas) in some cases for large search time budgets and bigger search radii. Accordingly, using both algorithms selectively appears to be a reasonable deployment strategy in practice.

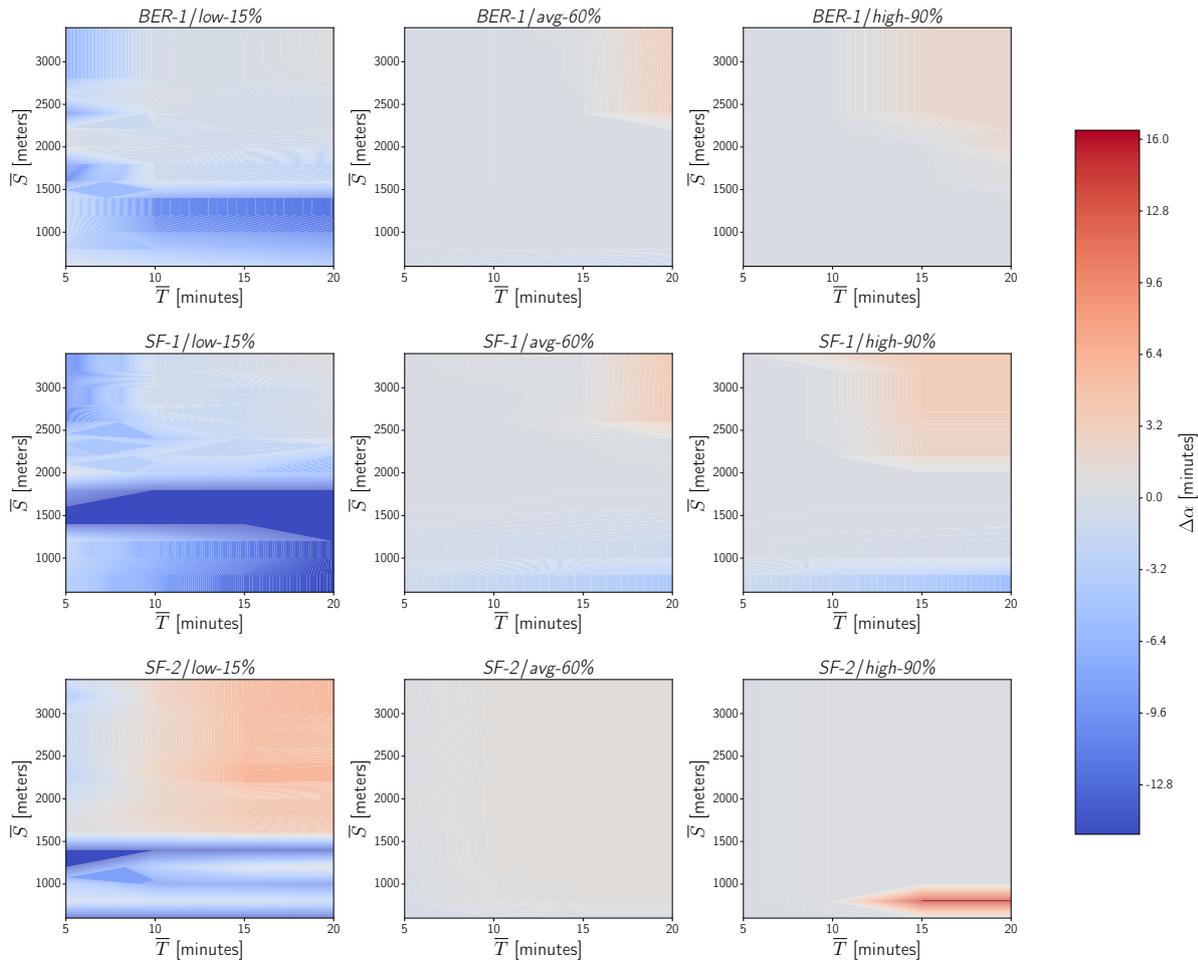


Figure 8.: Extensive heuristic comparison for problem variant $\neg W/\neg C$

Each subplot shows $\Delta\alpha = \alpha^{LH} - \alpha^{RO}$ (with α^{LH} , resp. α^{RO} being the solution cost for LH, resp. RO) as a function of \bar{T} and \bar{S} , where we limit LH computational times to 1 second. Each subplot corresponds to one of the 9 scenarios resulting from the combination of each area ($SF-1$, $BER-1$, $SF-2$) with each availability distribution ($low-15\%$, $avg-60\%$, $high-90\%$). Over all subplots, availability increases from left to right and station density increases from top to bottom.

5.2. Extended analysis

In the following, we analyze the sensitivity of our algorithms towards parameter and design decisions. We first analyze the algorithms' sensitivity towards the penalty cost $\bar{\beta}$ for $\neg W$ problem variants, before we study the impact of a time dependent recovery function, substantiate the heuristic dominance decisions and study the impact of additional charging times due to battery depletion during the search.

Termination penalty

We limit the discussion of termination penalty sensitivities to low charging station availability instances as these appear to be the most sensitive. Apparently, analyzing the search cost α does not allow for a meaningful interpretation of $\bar{\beta}$ sensitivities as α naturally increases with increasing $\bar{\beta}$. To circumvent this issue we decompose α into user-relevant metrics (see Appendix C), and analyze the computational time, the expected search time t_s and the probability $\bar{\rho}$ that the search unsuccessfully terminates, averaged over six $low-15\%$ -instances for problem variant $\neg W/\neg C$ (9a) and $\neg W/C$ (9b) in Figure 9.

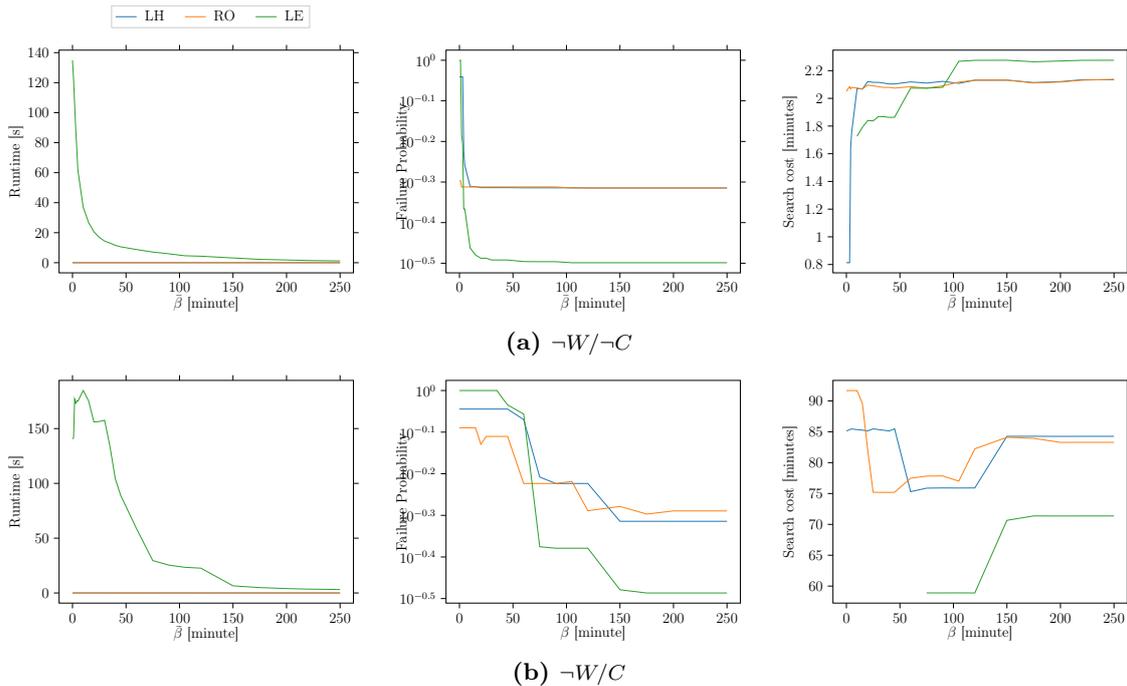


Figure 9.: Impact of $\bar{\beta}$ on averaged computational time t^s and failure rate for the *low-15%* instances

As can be seen, we observe a goal conflict between the expected search time and the search’s success rate: with increasing β we obtain better success rates at the price of higher expected search times. We note that one must choose $\beta \geq 30$ minutes for problem variant $\neg W/C$, as otherwise the cost for not visiting any station is lower than the cost for visiting at least one station (i.e., minimal charging time). We observe that the β values which are necessary to obtain the best possible success rate are significantly higher (120 minutes respectively 200 minutes for $\neg W/\neg C$ and $\neg W/C$) than this lower bound.

While the computational times of LE significantly decrease with increasing β , the computational times of LH and RO remain insensitive to changes in β for low charging station availability scenarios. However, additional analysis show that LH computational times increase with increasing β in high charging station availability scenarios, such that varying β for risk-adverse searches in a real-world implementation (i.e., choosing a higher β) should be chosen with respect to the computational overhead for LH.

Time-dependent recovery function

To analyze the impact of considering time-dependent recovering probabilities, we compare the objective values obtained in our main model (without recovering) with the model introduced in Section 3.4 (with recovering). In the former we compute the search path with a persistent charging station occupancy but calculate the objective value α considering time-dependent recovery for a fair comparison of both models. Based on preliminary studies, we set the average occupancy time to two hours, $\frac{1}{\mu} = 120$ minutes.

Table 11 compares the value of α for both models. As can be seen no significant difference exists between the initial model and the updated model at the exception of smaller areas with large time budget, particularly in case of low station availability. This is the only particular case where visiting

Table 11.: Potential solution improvement for the time-dependent probability recovery function for problem variant $\neg W/\neg C$

		<i>low-15%</i>				<i>avg-60%</i>			
		LH		RO		LH		RO	
\bar{T}	\bar{S}	α^{ref}	α^{new}	α^{ref}	α^{new}	α^{ref}	α^{new}	α^{ref}	α^{new}
5	800	25.3	25.1	28.5	28.7	1.41	1.41	1.55	1.54
5	2000	13.9	13.9	19.9	19.8	1.34	1.34	1.34	1.34
5	3400	9.63	9.52	19.9	19.8	1.34	1.34	1.34	1.34
10	800	23.9	24.8	28.5	29.3	1.35	1.35	1.57	1.55
10	2000	6.84	6.84	15.2	15.2	1.22	1.22	1.23	1.22
10	3400	3.22	3.22	8.60	4.65	1.22	1.22	1.22	1.22
15	800	23.9	23.4	28.5	28.6	1.35	1.32	1.55	1.53
15	2000	4.23	4.23	8.95	6.84	1.22	1.22	1.22	1.22
15	3400	2.44	2.44	3.11	2.86	1.22	1.22	1.22	1.22
20	800	23.9	17.2	28.5	27.8	1.35	1.33	1.55	1.52
20	2000	3.56	3.52	8.95	9.21	1.22	1.22	1.22	1.22
20	3400	2.41	2.41	2.95	2.83	1.22	1.33	1.22	1.22

The table compares for *BER-1* combined with *low-15%* and *avg-60%* the objective value obtained in the updated setting (α^{new}) and the initial setting (α^{ref}). The table excludes *high-90%* results as these do not show any deviations. Significant differences are shown in bold characters.

stations multiple times within the time budget might be worth an extra detour, as one could expect at least one of the observed stations to be freed.

Relaxed dominance criteria

To design our labeling heuristic, we relaxed the dominance check of LE as described in Section 3. In the following, we substantiate the design decision from Section 3. Here, we identify each possible variant of the dominance check with a boolean quintuple that signifies whether an equation (3.21, 3.22, 3.23, 3.24, 3.25 resp. 3.26) is active (= 1) or not (=0) in the respective dominance check, e.g., quintuple (1,0,1,0,0) identifies the dominance check variant in which only equations 3.21 and 3.23 are active.

Figure 10 shows the trade-off between the optimality gap and the computational times for all dominance criterion and problem variants. As can be seen, the (heuristic) dominance criterion as chosen in Section 3 – (1,1,0,0,0) – yields the lowest computational times possible to achieve the best possible solution quality among all heuristic dominance criteria for problem variants $\neg W/\neg C$, $W/\neg C$, and $\neg W/C$. For W/C , (1,1,0,0,1) yields the best trade-off by slightly decreasing computational times obtained with (1,1,0,0,0) but selecting (1,1,0,0,0) allows the best possible generic implementation for LH.

Battery depletion

For the following experiments, we consider a flat topography and base additional charging time calculations due to battery depletion on the technical characteristics of a Renault Zoe (battery: Z.E.50 [52 kWh], engine: R110/R135, (Renault 2020)). Furthermore, we assume that the battery level remains at minimum at 20% of its maximum capacity to account for safety considerations or driver anxiety. We approximate the extra charging time based on a linear charge curve since the additional depletion

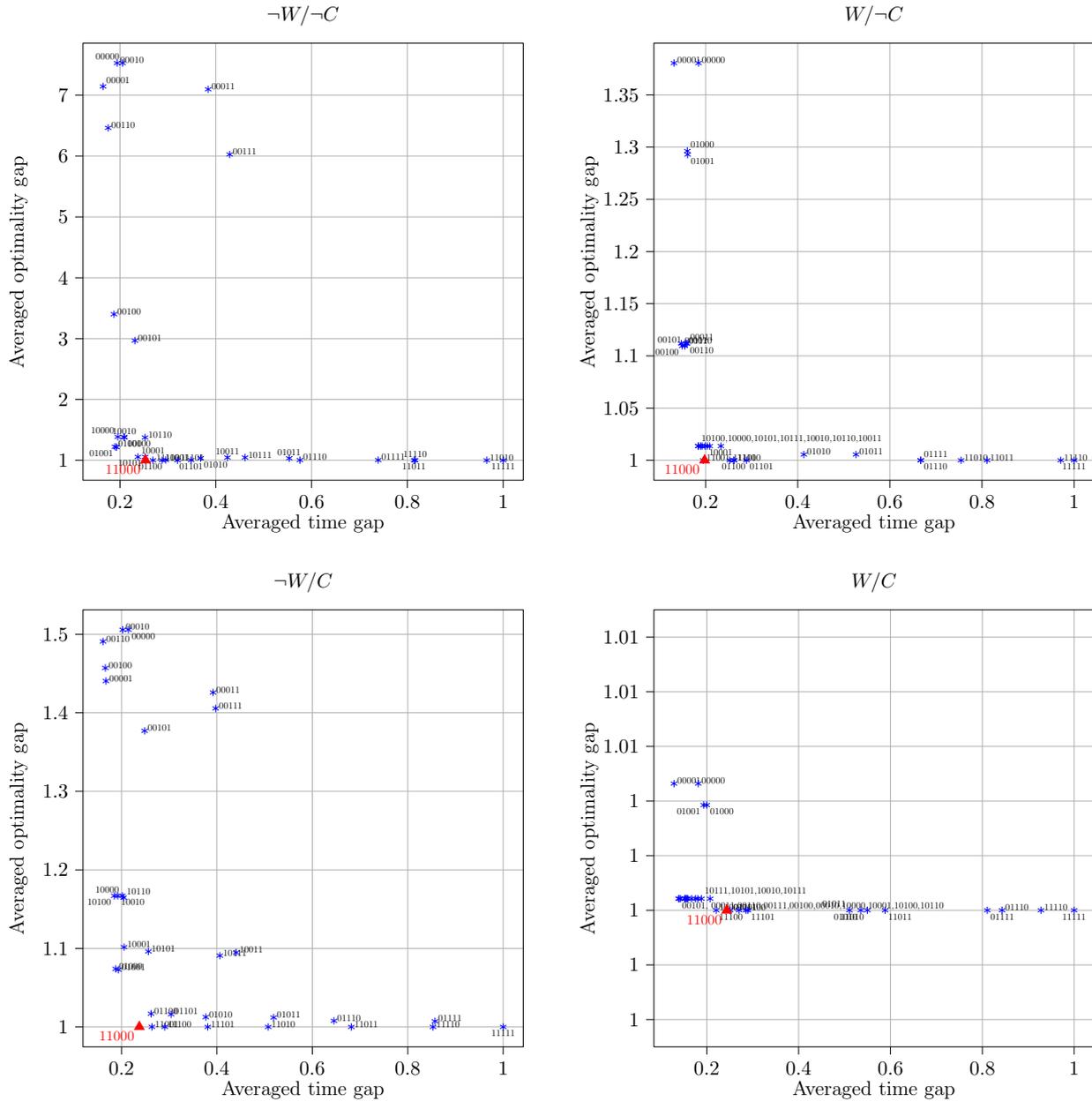


Figure 10.: Comparison of heuristic dominance criteria for all problem variants

The Figure shows the averaged optimality gap $g_\alpha = \sum_i \alpha_i / \alpha_i^{\text{opt}}$ as a function of the averaged computational time gap $g_t = \sum_i t_i / t_i^{\text{opt}}$ for each possible heuristic criterion for all variants. For each variant, both values are averaged over 16 instances corresponding to *BER-1* and *SF-1* combined resp. with *low-15%*, *avg-60%* and *high-90%* for $\bar{S} \in [1200, 1400, 1600, 1800]$ and fixed $\bar{T} = 10$. Red triangles show results for our selected dominance criteria.

adds only a limited amount to a (partial) recharge and remains in the linear part of the overall charge curve. To calculate energy consumption, we assume a constant speed of $v = 50\text{km/h}$, which remains a worst case estimate in an urban context. We define L_c as the expected time to charge the battery from its initial state to a full state at station c . Then, the depleted amount of energy after t (additional) minutes of driving results to $\delta b_c(v) = \frac{t}{T_a(v)} \times 52$.

In this setting, our results show that the impact of additional charging times due to an extended search and the resulting battery depletion is very limited in urban areas. For LH, we observe an average objective value change of 0.02%, which amounts to a maximum deviation of 1.5% for single instances.

The RO algorithm appears to be even less sensitive to the changed objective and shows no significant differences.

6. Conclusion and outlook

In this paper, we studied charging station search algorithms for stochastic environments, motivated by real-world applications in today’s navigation system applications. We introduced the underlying problem as a finite horizon MDP that covers several real-world problem variants. In this setting, we aim to find cost minimal search paths for an individual driver. We developed three solution algorithms: an exact labeling algorithm as well as a heuristic labeling algorithm and a rollout heuristic. We benchmarked these algorithms using an extensive real-world case study with instances for the cities of San Francisco and Berlin. Our results show that the heuristic algorithms allow for a significant speed-up compared to the exact algorithm at a price of a reasonable performance loss. Moreover, we show that our algorithms significantly improve a driver’s success to find a free charging station compared to myopic and greedy search approaches. This is in particular the case if the number of free charging stations in the search area is scarce. In this case, our algorithms reduce a driver’s search time by up to 44%.

In future work, we aim to leverage this work to study the impact of coordination and information sharing between multiple drivers. By so doing, we can study the impact of additional coordination that reduces the amount of uncertainty in the system. Moreover, studying the charging station search problem from a system perspective with a perfect information setting may yield an interesting upper bound that allows for an improved assessment of the solution quality of our algorithms in a stochastic setting.

Acknowledgements

This work was partially funded by the German Federal Ministry for Economic Affairs and Energy within project iMove (01ME16003B). The authors would like to thank Marko Rosenmüller and Andreas Linscheid for their support in the Berlin availability study.

References

- Amazon (2019). Amazon co-founds the climate pledge, setting goal to meet the paris agreement 10 years early. <https://www.businesswire.com/news/home/20190919005609/en/> (last accessed: 2019-12-14). Last accessed: 11.27.2019.
- Arndt, T., Hafner, D., Kellermeier, T., Krogmann, S., Razmjou, A., Krejca, M. S., Rothenberger, R., & Friedrich, T. (2016). Probabilistic routing for on-street parking search. In *24th Annual European Symposium on Algorithms (ESA 2016)* (pp. 6:1–6:13). volume 57.
- Bensasson, B. (2019). Pivot power acquisition press release. https://www.edf.fr/sites/default/files/20191104pr_edf_group_acquires_pivot_power_certified.pdf_ang_0.pdf. Last accessed: 11.27.2019.
- Bonges, H. A., & Lusk, A. C. (2016). Addressing electric vehicle (ev) sales and range anxiety through parking layout, policy and regulation. *Transportation Research Part A: Policy and Practice*, 83, 63–73.
- DHL (2019). Dhl electro mobility press release. <https://www.dpdhl.com/en/media-relations/specials/electromobility.html>. Last accessed: 11.27.2019.
- Fröhlich, K. (2019). Bmw electro mobility press release. <https://www.press.bmwgroup.com/global/article/detail/T0302391EN/bmw-group-continues-to-expand-charging-infrastructure-for-electrified-vehicles>. Last accessed: 11.27.2019.
- Goodson, J. C., Thomas, B. W. T., & Ohlmann, J. W. (2017). A rollout algorithm framework for heuristic solutions to finite-horizon stochastic dynamic programs. *European Journal of Operational Research*, 258, 216–229.

- Guo, Q., & Wolfson, O. (2018). Probabilistic spatio-temporal resource search. *GeoInformatica*, 22, 75–103.
- Jafari, E., & Boyles, S. D. (2017). Multicriteria stochastic shortest path problem for electric vehicles. *Networks and Spatial Economics*, 17, 1043–1070.
- Jossé, G., Schmid, K. A., & Schubert, M. (2015). Probabilistic resource route queries with reappearance. In *Proceedings of the 18th International Conference on Extending Database Technology* (pp. 445–456).
- Kullman, N., Goodson, J. C., & Mendoza, J. E. (2017). Electric vehicle routing with uncertain charging station availability & dynamic decision making. In *INFORMS Transportation and Logistics Society Triennial Conference*.
- Open Charge Map contributors (2019). Charging stations retrieved from Open Charge Map. <https://api.openchargemap.io/v3/referencedata/>.
- Renault (2020). Autonomie, batterie et recharge, nouvelle renault zoe. <https://www.renault.fr/vehicules-electriques/zoe/batterie-recharge.html>.
- Schmoll, S., & Schubert, M. (2018). Dynamic resource routing using real-time dynamic programming. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (pp. 4822–4828).
- Sweda, T. M., Dolinskaya, I. S., & Klabjan, D. (2017). Adaptive routing and recharging policies for electric vehicles. *Transportation Science*, 51, 1326–1348.

Appendix A Proofs

Proof of Proposition 1. Let π be the policy associated to sequence $C = (c_0, c_1, \dots, c_n)$. We consider state $x_k = (c_k, 0)$ with charging station c_k not being available and $c_k \neq c_n$. Thus, for $i < n$, $\prod_{j=i}^i (1 - p_{c_j}) = 1$. For $i = n$, $\prod_{j=i}^i (1 - p_{c_j}) = 1 - p_{c_n}$.

We then introduce F as follows

$$\begin{aligned}
 F^\pi(x_k) &= \prod_{i=k}^n (1 - \tilde{p}_{c_i}(C_{[k:i-1]})) \beta_{c_n} + \sum_{i=k}^{n-1} \left[t_{c_i, c_{i+1}} \prod_{j=k}^i (1 - \tilde{p}_{c_j}(C_{[k:j-1]})) \right] \\
 &+ \sum_{i=k}^n \left[\gamma_{c_i} \tilde{p}_{c_i}(C_{[k:i-1]}) \prod_{j=k}^{i-1} (1 - \tilde{p}_{c_j}(C_{[k:i-1]})) \right].
 \end{aligned} \tag{A.1}$$

We notice that $F^\pi(x_n) = (1 - p_{c_n}) \beta_{c_n} + p_{c_n} \gamma_{c_n} = V^\pi(x_n)$.

We now show that F fulfills the recursive definition of the policy specific cost function and by recursion that $F = V$. From state $x_k = (C_{[0,k]}, 0)$, we let the cost for being in next state $x_{k+1} = (C_{[0,k+1]}, 0)$ and seek to express $F^\pi(x_k)$ as a function of $F^\pi(x_{k+1})$ to fulfill the recursive definition 3.5.

$$\begin{aligned}
 F^\pi(x_{k+1}) &= \prod_{j=k+1}^n (1 - p_{c_j}) (\beta_{c_n}) + \sum_{i=k+1}^{n-1} [t_{c_i, c_{i+1}} \prod_{j=k+1}^i (1 - p_{c_j})] \\
 &+ \sum_{i=k+1}^n [\gamma_{c_i} p_{c_i} \prod_{j=k+1}^{i-1} (1 - p_{c_j})]
 \end{aligned} \tag{A.2}$$

$$\begin{aligned}
 F^\pi(x_k) &= (1 - p_{c_k}) \prod_{j=k+1}^n (1 - p_{c_j}) (\beta_{c_n}) + t_{c_k, c_{k+1}} + (1 - p_k) \sum_{i=k+1}^{n-1} [t_{c_i, c_{i+1}} \prod_{j=k+1}^i (1 - p_{c_j})] \\
 &+ p_{c_k} \gamma_{c_k} + (1 - p_{c_k}) \sum_{i=k}^n [\gamma_{c_i} p_{c_i} \prod_{j=k}^{i-1} (1 - p_{c_j})]
 \end{aligned} \tag{A.3}$$

$$F^\pi(x_k) = t_{c_k, c_{k+1}} + (1 - p_{c_k}) F^\pi(x_{k+1}) + p_{c_k} \gamma_{c_k}$$

Accordingly, F fulfills the recursive definition 3.5 for $w = 0$, which concludes the proof. \square

Proof of Proposition 2. We consider two simple search sequences (c) and (c, c') extended with the same visit sequence $C = (c_0, \dots, c_n)$. Let $C' = (c) \circ C$, resp. $C'' = (c, c') \circ C$, associated to policies π' , resp. π'' . Let $t_{c,c_0} = t_{c,c'} + t_{c',c_0}$ and let c' be a direct neighbor of c (i.e., there is no station d such that $t_{c,c'} = t_{c,d} + t_{d,c'}$) and let c_0 not be a direct neighbor.

We now show that from the considered state $x_0 = ((c), 0)$, visiting c' before c_0 is always better than straightforwardly visiting c_0 . We get

$$V^{\pi'}((c), 0) = t_{c,c_0} + (1 - \tilde{p}_{c_0})V^{\pi'}((c, c_0), 0),$$

$$V^{\pi''}((c), 0) = t_{c,c'} + (1 - \tilde{p}_{c'})t_{c',c_0} + (1 - \tilde{p}_{c'})(1 - \tilde{p}_{c_0})V^{\pi''}((c, c', c_0), 0)$$

and distinguish two cases:

Case 1 ($c' \notin C$): In this case, the valuation of any unexplored station after c_0 does not depend on preceding visits in the respective sequence, i.e., $V^{\pi''}((c, c_0), 0) = V^{\pi''}((c, c', c_0), 0)$. Given that $(1 - \tilde{p}_c) \leq 1$, we straightforwardly obtain $V^{\pi''}((c), 0) \leq V^{\pi'}((c), 0)$.

Case 2 ($c' \in C$): In this case, the valuation of any unexplored station after c_0 depends on preceding visits in the respective sequence and we obtain the following cost expansions, visiting c' at position k :

For path C' , c' is visited for the first time such that $\tilde{p}_{c_k} = p_{c'}$ and we get

$$V^{\pi'}(C', 0) = \prod_{j=0}^n (1 - \tilde{p}_{c_j})(\bar{\beta}) + \sum_{i=0}^{k-1} t_{c_i, c_{i+1}} \prod_{j=0}^i (1 - \tilde{p}_{c_j}) + \sum_{i=k}^{n-1} [t_{c_i, c_{i+1}} (1 - \tilde{p}_{c'}) \prod_{j=0, j \neq k}^i (1 - \tilde{p}_{c_j})] \quad (\text{A.4})$$

For path C'' , c' is visited for the second time such that $\tilde{p}_{c_k} = 0$ and we get

$$(1 - \tilde{p}_{c'})V^{\pi''}(C'', 0) = (1 - \tilde{p}_{c'}) \prod_{j=0}^n (1 - p_{c_j})(\bar{\beta}) + (1 - \tilde{p}_{c'}) \sum_{i=0}^{k-1} [t_{c_i, c_{i+1}} \prod_{j=0}^i (1 - p_{c_j})] + \sum_{i=k}^{n-1} [t_{c_i, c_{i+1}} (1 - \tilde{p}_{c'}) \prod_{j=0, j \neq k}^i (1 - p_{c_j})] \quad (\text{A.5})$$

Since $(1 - \tilde{p}_{c'}) \leq 1$, we have

$$(1 - \tilde{p}_{c'})V^{\pi''}((c, c', c_0), 0) \leq V^{\pi'}((c, c_0), 0)$$

and consequently

$$V^{\pi''}(x_0) \leq V^{\pi'}(x_0).$$

In both cases, π'' is preferred over π' (thus C'' over C'), such that candidate stations can be restricted to neighbor stations only, which concludes the proof. \square

Appendix B Problem Complexity

Proposition 3. *The SCPS problem is NP-hard, even with metric travel times.*

Proof. We show hardness through reduction from the traveling salesman problem (TSP) with metric and integer travel times. The decision problem variant of the TSP can be defined as follows: we consider a set \mathcal{C} of n sites and travel times $t_{c,c'} \in \mathbb{N}$ between these. Travel times are bounded $1 \leq t_{c,c'} \leq \Delta$ for all $c, c' \in \mathcal{C}$. We are asked for a tour (i.e., a Hamiltonian path) $c_0, \dots, c_n = c_0$ whose length satisfies $\sum_{i=0}^{n-1} t_{c_i, c_{i+1}} \leq \theta$ for a given θ . Given that travel times are integer, we can assume w.l.o.g. $\theta \in \mathbb{N}$. Further, we assume that triangle inequality holds, i.e., $t_{c,c'} + t_{c',c''} \geq t_{c,c''}$ for all $c, c', c'' \in \mathcal{C}$. We note that hardness for this restricted metric case implies hardness for the generic case as well.

Step 1: We construct an instance for the station search from the TSP instance as follows: We select an arbitrary vertex $s \in \mathcal{C}$ and designate it as start vertex $c_0 := s$ for the search. We then create a duplicate s' in the same location ($t_{s,s'} = 0$), which serves as the termination vertex. Let q be an arbitrary value satisfying

$$\left(1 - \frac{1}{\Delta(n+1)}\right)^{\frac{1}{n-1}} \leq q < 1. \quad (\text{B.1})$$

We parameterize the search as follows: All vertices c have an availability of $p_i := 1 - q$ without recovery. There is no penalty for successful charging ($\gamma_i := 0 \forall i$). For unsuccessfully terminating at s' , the penalty is

$$\beta_{s'} := \frac{2\Delta}{q(1-q)} + 1. \quad (\text{B.2})$$

For all other vertices $c \neq s'$ the penalty is

$$\beta_c := \frac{1}{q^{n+1}}(\beta_{s'} + n\Delta) + 1. \quad (\text{B.3})$$

Now for any search path $C = (c_0, \dots, c_k)$ that does not visit any vertex multiple times, it holds that

$$\begin{aligned} \alpha(C) &= \left(\prod_{i=0}^k (1 - p_{c_i})\right) \beta_{c_k} + \sum_{i=0}^{k-1} t_{c_i, c_{i+1}} \prod_{j=0}^i (1 - p_{c_j}) + \sum_{i=0}^k \gamma_{c_i} p_{c_i} \prod_{j=0}^{i-1} (1 - p_{c_j}) \\ &= q^{k+1} \beta_{c_k} + \sum_{i=0}^{k-1} q^{i+1} t_{c_i, c_{i+1}}. \end{aligned} \quad (\text{B.4})$$

Step 2: We now claim that the TSP instance possesses a solution with cost at most $\theta \in \mathbb{N}$ if, and only if, the station search admits a search path C with $\alpha(C) \leq q\theta + q^{n+1}\beta_{s'}$. This is done by transforming solutions between the two problems and carefully mapping their objective values.

For the first direction, we assume that a TSP tour is given. We convert it to a search path $C = (c_0, \dots, c_n)$ by cutting at s such that $c_0 = s$ and $c_n = s'$. Then

$$\alpha(C) = q^{n+1}\beta_{s'} + \sum_{i=0}^{n-1} q^{i+1} t_{c_i, c_{i+1}} \leq q^{n+1}\beta_{s'} + q \sum_{i=0}^{n-1} t_{c_i, c_{i+1}} \leq q^{n+1}\beta_{s'} + q\theta.$$

Vice versa, we assume that we are given a search path P with $\alpha(P) \leq q\theta + q^{n+1}\beta_{s'}$. Then for an optimal search path $C = (c_0, \dots, c_k)$ it holds that $\alpha(P) \leq \alpha(C) \leq q\theta + q^{n+1}\beta_{s'}$. Given metric travel

times and no recovery, we can assume that C does not visit any vertex more than once. We construct a tour through the following observations:

1. C visits s' : Assume it does not. Let C' be C extended by ending at s' . Then

$$\begin{aligned}\alpha(C') &= \alpha(C) - q^{k+1}\beta_{c_k} + q^{k+2}\beta_{s'} + q^{k+1}t_{c_k, s'} \\ &= \alpha(C) - q^{k+1}(\beta_{c_k} - q\beta_{s'} - t_{c_k, s'}) \stackrel{(*)}{<} \alpha(C)\end{aligned}\tag{B.5}$$

using in (*) that $\beta_{c_k} > q\beta_{s'} + \Delta$ by (B.3). This contradicts the optimality of C .

2. C visits s' last: Assume it does not. We obtain C' from C by moving s' to the end. Then it holds that

$$\begin{aligned}\alpha(C) &\geq q^{k+1}\beta_{c_k} \geq q^{n+1}\beta_{c_k} \text{ and} \\ \alpha(C') &\leq q^{k+1}\beta_{s'} + \sum_{i=0}^{k-1} q^{i+1}\Delta \leq \beta_{s'} + n\Delta.\end{aligned}\tag{B.6}$$

Given that $q^{n+1}\beta_{c_k} > \beta_{s'} + n\Delta$ (B.3), it holds that $\alpha(C) > \alpha(C')$, contradicting optimality.

3. C visits every vertex: Assume $C = (c_0, \dots, c_{k-1}, s')$ omits some vertex c' . Let $C' = (c_0, \dots, c_{k-1}, c', s')$. Then

$$\begin{aligned}\alpha(C) - \alpha(C') &= \left(q^k t_{c_{k-1}, s} + q^{k+1}\beta_{s'}\right) - \left(q^k t_{c_{k-1}, c'} + q^{k+1}t_{c', s'} + q^{k+2}\beta_{s'}\right) \\ &= q^k(t_{c_{k-1}, s} - t_{c_{k-1}, c'} - qt_{c', s'} + (q - q^2)\beta_{s'}) \\ &\geq q^k(-2\Delta + q(1 - q)\beta_{s'}) \stackrel{\text{(B.2)}}{>} 0,\end{aligned}\tag{B.7}$$

again contradicting optimality.

4. It is clear now that C corresponds to a TSP tour, by identifying s with s' . It holds that

$$\alpha(C) = \sum_{i=0}^{n-1} q^{i+1}t_{c_i, c_{i+1}} + q^{n+1}\beta_{s'} \leq q\theta + q^{n+1}\beta_{s'}.\tag{B.8}$$

Assume the tour would violate the threshold, i.e., $\sum_{i=0}^{n-1} t_{c_i, c_{i+1}} > \theta$. Given integrality, the length then is at least $\theta + 1$. It follows that

$$\begin{aligned}\sum_{i=0}^{n-1} q^{i+1}t_{c_i, c_{i+1}} &\geq \sum_{i=0}^{n-1} q^n t_{c_i, c_{i+1}} = q \sum_{i=0}^{n-1} q^{n-1} t_{c_i, c_{i+1}} \\ &\stackrel{\text{(B.1)}}{\geq} q \sum_{i=0}^{n-1} \left(1 - \frac{1}{\Delta(n+1)}\right) t_{c_i, c_{i+1}} = q \left[\sum_{i=0}^{n-1} t_{c_i, c_{i+1}} - \sum_{i=0}^{n-1} \frac{t_{c_i, c_{i+1}}}{\Delta(n+1)} \right] \\ &\geq q \left[\theta + 1 - \sum_{i=0}^{n-1} \frac{1}{n+1} \right] > q\theta.\end{aligned}\tag{B.9}$$

This contradicts (B.8), thereby proving that $\sum_{i=0}^{n-1} t_{c_i, c_{i+1}} \leq \theta$.

□

Appendix C Search cost decomposition

In this section, we show by proving Proposition 4 that the total search cost $\alpha(\pi)$ as derived in Section 2 can be expressed as a function $t(\pi)$, $\rho(\pi)$ and an additional quantity $t^s(\pi)$ representing the expected time to find a station assuming at least one station is available among C , the visits sequence associated with π . We derive the quantity $t_s(\pi)$ based on the work of Arndt et al. (2016) that describes $t^s(\pi)$ as a sum of the partial accumulated driving time to a station (parking spot in their work) weighted by the probability the driver charges exactly at this station.

Proposition 4. *Cost α can be decomposed to exhibit $t^s(\pi)$ as follows,*

$$\alpha(\pi) = t^s(\pi) \cdot \rho(\pi) + \bar{\rho}(\pi) \cdot (t(\pi) + \beta_{c_n}). \quad (\text{C.1})$$

Proof. We recall that $\alpha(\pi) = A(\pi) + \bar{\rho}(\pi) \cdot \beta_{c_n}$ (cf. Equation 3.13). For the sake of conciseness, we simplify the notation for the remainder of this proof as follows: $C = (0, 1, \dots, n)$ such that $t_{k,k+1} = t_{c_k, c_{k+1}}$, $\bar{\rho}_k = \prod_{i=0}^k \bar{p}_i$. We let $\rho_n = \rho(\pi)$, $A_n = A(\pi)$, $t_n = t(\pi)$, $t_n^s = t^s(\pi)$. We then define t^s based on Arndt et al. (2016) as

$$t_n^s = \frac{\sum_{k=0}^{n-1} \bar{\rho}_{k-1} p_k (t_k + \gamma_k)}{\rho_n}.$$

We now note that $\bar{\rho}_k p_k$ represents the probability of station k being available when visited, given that no station in $i \in (0, \dots, k-1)$ was available.

We then introduce the quantity $B_n = \sum_{k=0}^{n-1} \bar{\rho}_{k-1} p_k (t_k + \gamma_k)$ such that $t_n^s \cdot \rho_n = B_n$ and note that to prove C.1, it is sufficient to show

$$A_n = B_n + \bar{\rho}_n \cdot t_n, \quad (\text{C.2})$$

which follows by recursion:

Step 1: For $n = 0$, C.1 holds : $A_0 = t_{0,1}$ and $B_0 = p_1 t_{0,1} + \bar{p}_1 t_{0,1} = A_0$.

Step 2: We assume that C.1 holds and show that $A_{n+1} = B_{n+1} + \bar{\rho}_{n+1} \cdot t_{n+1}$ holds, too:

$$\begin{aligned} A_{n+1} &= A_n + \bar{\rho}_n \cdot (t_{n,n+1} + p_{n+1} \gamma_{n+1}) \\ B_{n+1} &= B_n + \bar{\rho}_n p_{n+1} \cdot (t_{n+1} + \gamma_{n+1}) \end{aligned} \quad (\text{C.3})$$

Given C.1,

$$\begin{aligned} A_{n+1} &= B_n + \bar{\rho}_n \cdot t_n + \bar{\rho}_n (t_{n,n+1} + p_{n+1} \gamma_{n+1}) \\ \Leftrightarrow A_{n+1} &= B_n + \bar{\rho}_n \cdot t_n + \bar{\rho}_n (t_{n,n+1} + p_{n+1} \gamma_{n+1}) + \bar{\rho}_n p_{n+1} \cdot (t_{n+1} + \gamma_{n+1}) - \bar{\rho}_n p_{n+1} \cdot (t_{n+1} + \gamma_{n+1}) \end{aligned} \quad (\text{C.4})$$

From C.3 and C.4, we get :

$$\begin{aligned} A_{n+1} &= B_{n+1} + \bar{\rho}_n \cdot (t_n) + \bar{\rho}_n (t_{n,n+1} + p_{n+1} \gamma_{n+1}) - \bar{\rho}_n p_{n+1} \cdot (t_{n+1} + \gamma_{n+1}) \\ \Leftrightarrow A_{n+1} &= B_{n+1} + \bar{\rho}_n \cdot t_{n+1} - \bar{\rho}_n (p_{n+1}) t_{n+1} \\ \Leftrightarrow A_{n+1} &= B_{n+1} + \bar{\rho}_{n+1} \cdot t_{n+1} \end{aligned} \quad (\text{C.5})$$

This concludes the proof. \square

Appendix D Reduced action spaces results

Table 12 compares the cost α , the computational times, and the percentage share of instances that can be computed in less than 15000 seconds by each heuristic. Further, for the instances that can be solved to optimality within 15000 seconds, the table compares for both heuristics the averaged optimality gap and computational time gap.

Table 12.: Aggregated computational results over all tested instances for each problem variant

Graph setup		LH					RO				
Problem variant	Action space	g_t	g_α	\hat{t}	$\hat{\alpha}$	\hat{n}	g_t	g_α	\hat{t}	$\hat{\alpha}$	\hat{n}
$\neg W/\neg C$	<i>direct</i>	0.21	1.01	197	7.90	0.73	0.35	1.27	0.08	11.2	1.00
	<i>direct/restricted</i>	0.16	1.04	235	7.91	0.82	–	–	–	–	–
	<i>T^r-restricted</i>	0.17	1.00	221	7.84	0.74	–	–	–	–	–
	<i>complete</i>	–	–	–	–	–	0.23	1.13	0.58	9.54	1.00
$W/\neg C$	<i>direct</i>	0.20	1.00	72.7	2.48	0.83	0.20	1.03	0.03	2.52	1.00
	<i>direct/restricted</i>	0.16	1.02	16.7	2.49	1.00	–	–	–	–	–
	<i>T^r-restricted</i>	0.16	1.01	250	2.46	0.78	–	–	–	–	–
	<i>complete</i>	–	–	–	–	–	0.36	1.04	0.76	2.53	1.00
$\neg W/C$	<i>T^r-restricted</i>	0.17	1.01	311	48.6	0.84	0.18	1.12	0.31	53.1	1.00
	<i>complete</i>	0.16	1.01	486	48.4	0.84	0.19	1.12	0.38	53.6	1.00
W/C	<i>T^r-restricted</i>	0.26	1.00	120	34.5	0.83	0.31	1.00	0.88	34.5	1.00
	<i>complete</i>	0.20	1.00	100	34.5	0.83	0.33	1.00	1.05	34.5	1.00

Abbreviations hold as follows: g_α - averaged optimality gap over all tested instances with $g_\alpha = \frac{\alpha^{\text{heur}}}{\alpha^{\text{opt}}}$, g_t - averaged computational time gap with $g_t = \frac{t^{\text{heur}}}{t^{\text{opt}}}$, $\hat{\alpha}$ - averaged search cost α [min], \hat{t} - averaged computational time[s], \hat{n} - rate of instances that can be computed in less than 15000 seconds.

Preliminary results show that using the *complete* action space for problem variants $\neg W/\neg C$ and $W/\neg C$ with LH is computationally too heavy to be of any practical use, such that we restrict results to the other restricted action spaces. As can be seen, while *direct* only slightly helps saving computational times on average for $\neg W/\neg C$ and LH, it allows to solve 6% more instances within the allocated time for $W/\neg C$. For RO, in $W/\neg C$, results show a 96% decrease of the computational time. For $\neg W/C$, restricting the next station visits from the current location to the ones accessible in less than 5 minutes allows to save 36% of the computational times compared to *complete*. Accordingly, Table 13 shows the most appropriate action space for each heuristic and problem variant.

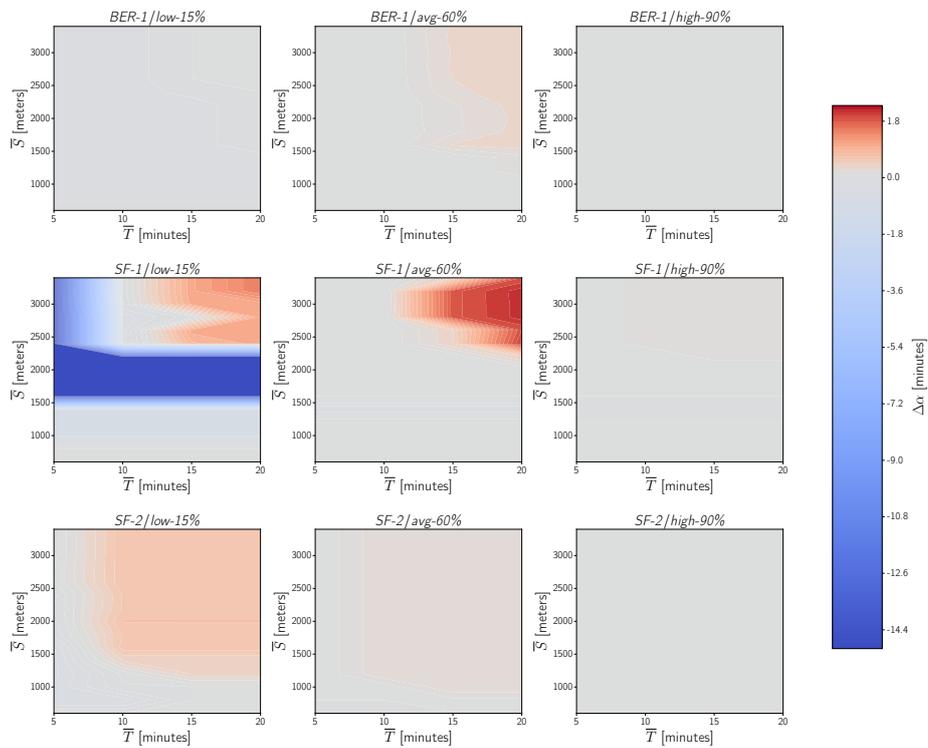
Table 13.: Best action space /heuristic combination per problem variant

Problem variant	LH	RO
$\neg W/\neg C$	<i>direct</i>	<i>complete</i>
$W/\neg C$	<i>direct</i>	<i>direct</i>
$\neg W/C$	<i>T^r-restricted</i>	<i>complete</i>
W/C	<i>complete</i>	<i>complete</i>

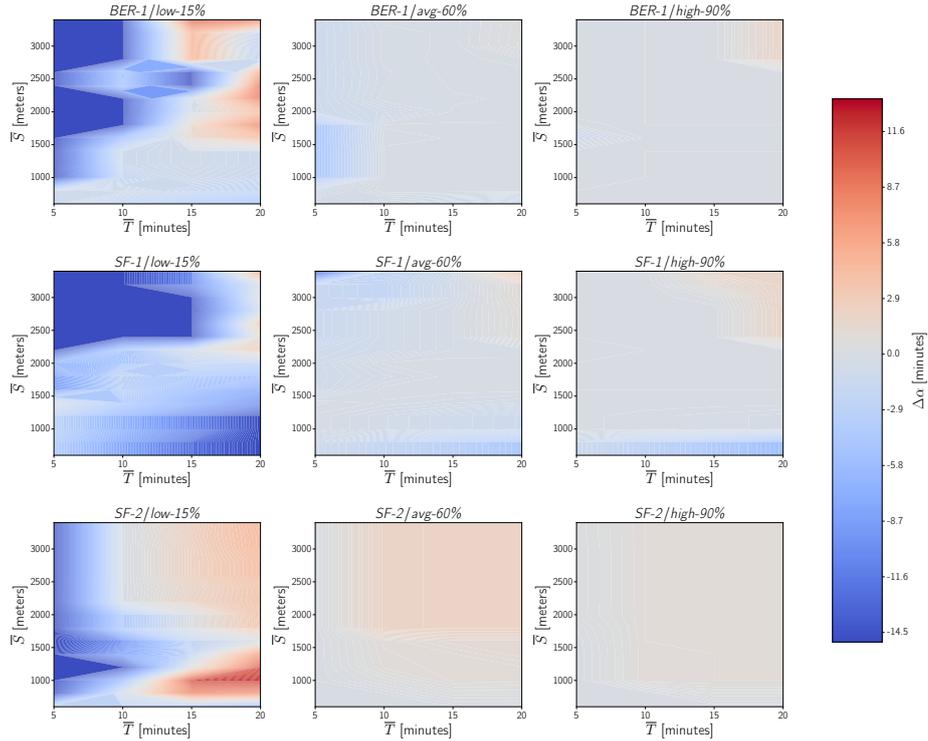
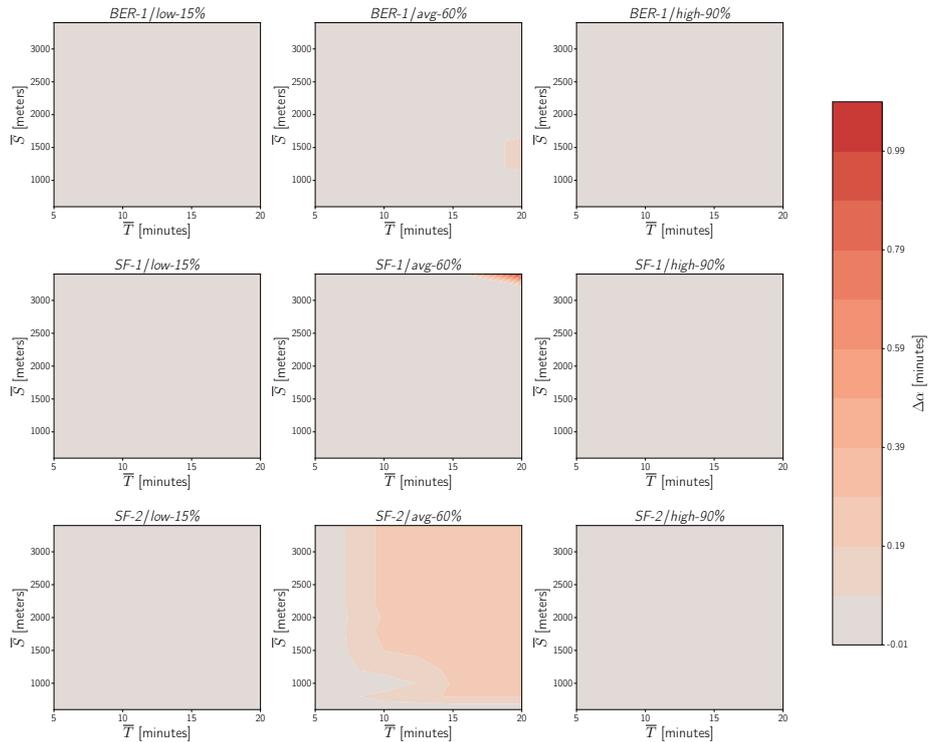
For all problem variants, the table shows for each heuristic (LH, RO) the graph setting to that provides the best trade-off between computational times and solution quality.

Appendix E Additional tractability analysis results

Figure 11 shows the extensive heuristic comparisons for the remaining problem variants $W/\neg C$, $\neg W/C$, W/C .



(a) Heuristic comparison for problem variant $W/\neg C$

(b) Heuristic comparison for problem variant $-W/C$ (c) Heuristic comparison for problem variant W/C Figure 11.: Extensive heuristic comparison for problem variants $W/-C$, $-W/C$, W/C

Each subplot shows $\Delta\alpha = \alpha^{LH} - \alpha^{RO}$ (with α^{LH} , resp. α^{RO} , the solution cost for LH, resp. RO) as a function of \bar{T} and \bar{S} , where we limit LH computational times to 1 second. Each subplot corresponds to one of the 9 scenarios resulting from the combination of each area ($SF-1$, $BER-1$, $SF-2$) with each availability distribution ($low-15\%$, $avg-60\%$, $high-90\%$). Over all subplots, availability increases from left to right and station density increases from top to bottom.