Optimal Charging Strategies for Electric Cars on Long Trips

Gerhard Huber, Klaus Bogenberger
Department of Traffic Engineering, University of Federal Armed Forces Munich
Werner-Heisenberg-Weg 39, 85577 Neubiberg, Germany
gerhard.huber@unibw.de, klaus.bogenberger@unibw.de

***Abstract*— The rather low range of electric vehicles makes it necessary to recharge on long-distance trips. A question, one can ask in this context, is where and how long should be charged in order to reach the destination as fast as possible, for example by avoiding rush hour, and certainly, i.e., without running out of energy. Traditional routing algorithms are not capable of providing this kind of information. In this paper, the task of finding such charging strategies is modeled as a bicriteria shortest path problem in a time-dependent network. It is discussed how the possibility to charge can be included in the graph-based model of a road network and how negative edge costs due to charging can be handled. Two algorithms are proposed for solving the described problem. The correctness of the first one is proven, whereas the second one is intended as a speed-up technique, which provides only approximative solutions.**

I. INTRODUCTION

During the last years, besides the traditional combustion engine, alternative driving concepts have emerged. Especially battery electric vehicles attract attention, as they are supposed to be environment-friendly and cheap in maintenance. However, low driving range and, as a consequence thereof, the so-called range-anxiety are often assumed to be crucial barriers for the establishment of pure electric cars. Covering long distances with electric vehicles, for example to go on holidays, is usually not even considered. One possibility to reduce range-anxiety and generate range certainty is to give adequate and reliable routing information. In case of distances that cannot be covered even with a fully charged battery, conventional navigation is not sufficient. On-trip information concerning the location of the next charging station available is one option for improvement. Another one, this paper is discussing, is pre-trip information. The idea is to provide, before the trip is started, a charging strategy for a given route. The charging strategy is intended similarly as a pit strategy: Instructions are given defining at which charging stations along the proposed route one has to charge and for how long. The goal is to reach the destination as fast as possible, but also certainly, i.e., without the risk of running out of energy. The fundamental idea in these considerations is the development of a charging strategy which allows avoiding rush hour traffic.

A. Scope of the Paper

There are many aspects that have to be covered for the development of a charging strategy: Traffic prediction, modeling the energy consumption of electric vehicles and analyzing factors influencing it, the charging behaviour and how the resulting optimization problem can be modeled and solved. In this paper the focus is set on modeling the optimization problem as a multicriteria, time-dependent point-to-point shortest path problem and on constructing an algorithm that leads to a time-optimal solution. Nevertheless, all mentioned topics, especially the energy consumption, are discussed in such a way that issues which are critical for modeling or solving the considered shortest path problem can be identified and their impact can be understood.

II. FINDING OPTIMAL PATHS IN TIME-DEPENDENT NETWORKS

Typically, the task of finding optimal paths is carried out on the basis of a database which contains information about a graph which again represents a road network. Formally, a (directed) graph $\overrightarrow{G}$ is a tuple $(V, \overrightarrow{E})$, where V denotes a set of vertices or nodes and $\overrightarrow{E} \subseteq V \times V$ is a set of edges connecting some of these nodes. In this paper, no multi-edges are allowed, i.e., for two nodes v_1 and v_2, at most one edge $(v_1, v_2) \in \overrightarrow{E}$ exists. It is also postulated that $v_1 \neq v_2$ for any edge $(v_1, v_2) \in \overrightarrow{E}$. Moreover, only finite graphs are considered, i.e., $\mid V \mid < \infty$. The edges, which basically represent road segments, possess features or attributes, e.g. the length of the road segment or its slope. In addition, to be able to optimize, a cost function $c : \overrightarrow{E} \longrightarrow \mathbb{R}$ assigning costs to the edges is necessary. These costs usually depend on the attributes of the respective edge. For instance, the cost function c_{dist} may assign the length in meter to road segments. A path on $\overrightarrow{G} = (V, \overrightarrow{E})$ is defined as a finite sequence of nodes $v_1, v_2, ..., v_n$, where for each pair of successive nodes (v_i, v_{i+1}) with $i \in \{1, 2, ..., n-1\}$ the corresponding edge exists, i.e., $(v_i, v_{i+1}) \in \overrightarrow{E}$. A path starting at a node v_1 and leading to a node v_2 is denoted as $v_1 - v_2-$path. Then, the problem of finding an optimal path from a node s to a node d on the directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$ w.r.t. a cost function c can be stated as:

$$\min c(P) \tag{1}$$

$$\text{subject to: } P \text{ is a } s\text{-}d\text{-path on } \overrightarrow{G}. \tag{2}$$

Note that $c(P)$ denotes the sum over the costs of all edges in path P.

A. Time-dependent Cost Functions

Finding optimal charging strategies will be modeled as a shortest path problem, i.e., one has to compute optimal

paths. Here, a path is called optimal if it leads to the lowest possible travel time. The cost function assigning travel times to road segments is denoted with c_T. As the traffic state clearly influences the time needed for passing a road segment and the traffic state itself strongly depends on time, c_T is time-dependent. When speaking of time-dependent costs, in this paper it is assumed that the costs for an edge are fixed at the time the edge is reached, i.e., if one wants to compute the costs for a path $P := [v_1, v_2, v_3]$ for a given starting time t_S and a given time-dependent cost function c, then

$$c(P, t_S) = c((v_1, v_2), t_S) + \tag{3}$$
$$c((v_2, v_3), t_S + c_T((v_1, v_2), t_S)). \tag{4}$$

Defining time-dependency in this way is denoted as frozen link model [16]. Clearly, the time costs c_T for any edge and any (starting) time cannot be negative:

$$c_T : \overrightarrow{E} \times \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}_{\geq 0} \tag{5}$$

Note that here it is assumed that the starting time t_S is also non-negative. However, only considering time costs is not realistic for developing a charging strategy. The limited range of electric vehicles makes it necessary to take also energy consumption into account. For instance, driving on highways leads usually to very high speeds and consequently such roads are interesting for optimizing travel times. On the other hand, high speeds lead to high air resistance and thus to high energy consumption for electric cars. In the case of long distances, this implies more stops for charging, which may be counter-productive for reducing the total travel time. Estimating energy consumption for certain road segments is not trivial, since the energy consumption of electric cars depends on many aspects. Interior factors like driving behaviour or vehicle type have influence as well as exterior factors as weather conditions, road features [6] and the current traffic situation [21]. Some of these factors are strongly time-dependent. As a result, it is necessary to define the cost function, which represents the energy consumption, as a time-dependent function, i.e.: $\tilde{c}_E : \overrightarrow{E} \times \mathbb{R}_{\geq 0} \longrightarrow \mathbb{R}$.

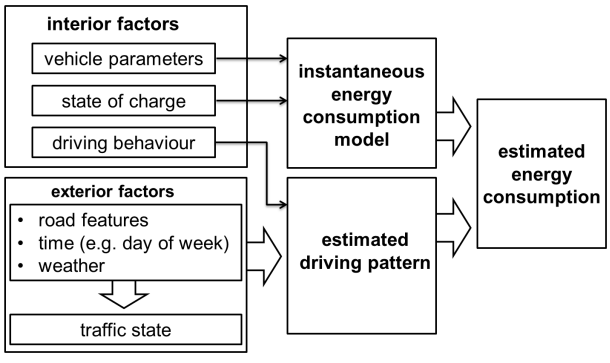


Fig. 1. Estimating the consumption of vehicles for road segments

A common approach to estimate the energy consumption of cars for certain road segments is based on physical consumption models [20]. These models allow the computation of the instantaneous fuel consumption by considering vehicle parameters, the slope of the road, the current speed and the current acceleration. Note that this also implies that for each car-type a new energy consumption cost function has to be constructed. Recently, such models were also developed for electric vehicles [13]. The most important difference between energy consumption models and traditional fuel consumption models is the possibility of gaining energy by electric vehicles due to recuperation. According to this last point, $\tilde{c}_E$ also assumes negative values, an aspect that will be of relevance later on. To get from the energy consumption model to an energy consumption $\tilde{c}_E(e, t)$ for a specific road segment $e \in \overrightarrow{E}$ at a certain time t, it is necessary to make assumptions about typical driving patterns for e at t. A driving pattern, on the other hand, depends on the driver, road features, the weather and the traffic state. Since the estimation of energy consumption along road segments is not in the scope of this paper, see [12] for more details concerning this topic.

Another important aspect, especially relevant for modeling charging processes, is that the energy consumption does also depend on the current state of charge. For example, if the battery is almost fully charged, then its charging behaviour impairs. The state of charge, denoted with SOC, results from the initial state of charge SOC_S, the precedent energy consumption and the energy capacity of the battery. As the state of charge is typically given as a percentage value, it only obtains values between 0 and 1. Here, it is distinguished between $\tilde{c}_E$, which are the energy consumption costs that does not take the current state of charge into account, and c_E, which quantifies the percentage change of the state of charge. Hence, for a given state of charge SOC, a given time t, the energy consumption costs $c_E : \overrightarrow{E} \times \mathbb{R}_{\geq} \times [0, 1] \longrightarrow [-1, 1]$ of an edge e are denoted by $c_E(e, t, SOC)$. As SOC can obtain only values between 0 and 1, it is postulated for any time $t > 0$, any $SOC \in [0, 1]$ and any $e \in \overrightarrow{E}$:

$$0 \leq SOC - c_E(e, t, SOC) \leq 1. \tag{6}$$

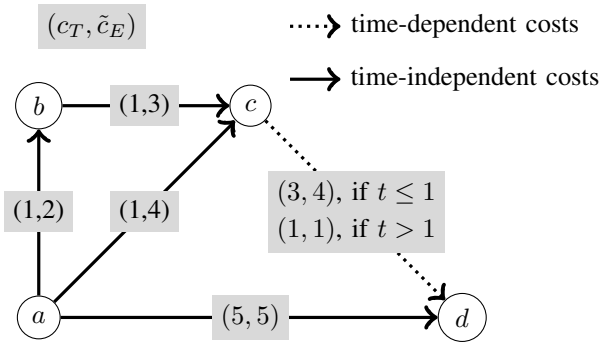
Condition 6 ensures that within the model one cannot charge (due to recuperation or charging at a charging station) more energy than the amount of energy which leads to a fully recharged battery. Additionally, it implies that if not enough energy is stored in the battery to drive to the end of a certain edge at a certain time, the resulting state of charge has to be set to 0. Consequently, the corresponding edge cannot be passed by the electric vehicle - at least not at the considered time with the given state of charge. This leads to the idea that only those paths are allowed during the route optimization, for which the state of charge does not get too low:

Definition 1. *Let a finite and directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$, two cost functions c_T, c_E as described above, a starting state of charge SOC_S, a minimal allowed state of charge $SOC_{min} > 0$ and a starting time t_S be given. Then, a path $P = (v_1, ..., v_n)$ on $\overrightarrow{G}$ is called* ***energy-secure*** *(w.r.t. SOC_{min}) if there is no node v_i of P with*

$$SOC_S - c_E((v_1, ..., v_i), t_S, SOC_S) < SOC_{min}. \tag{7}$$

An energy-secure path is actually a path where the state of charge never drops below the value SOC_{min}. Introducing SOC_{min} is necessary to be able to compensate uncertainties due to an inaccurate traffic prediction, or the unknown behaviour of the driver. The lower SOC_{min}, the higher the risk that mistakes of the energy cost estimation lead to an empty battery.

A small example for the computation of time-dependent cost functions is provided in Fig. 2. Here, the costs for



t_S	SOC_S	SOC_{min}	E_{max}
0	100 %	15 %	10

Fig. 2. Cost computation in time-dependent networks

all edges but for (c, d) are time-independent. The costs are displayed in the gray rectangles. The first entry within the brackets refers to the time costs c_T, the second entry to the energy consumption costs $\tilde{c}_E$. The costs $\tilde{c}_E$ for the edge (c, d) are given by:

$$\tilde{c}_E((c,d),t) := \begin{cases} 4, & t \leq 1; \\ 1, & else \end{cases} \tag{8}$$

The time costs $c_T((c, d), t)$ are defined by:

$$c_T((c,d),t) := \begin{cases} 3, & t \leq 1; \\ 1, & else \end{cases} \tag{9}$$

The values $SOC_S = 1.0$, $SOC_{min} = 0.15$ and the maximum energy capacity of the battery $E_{max} = 10$ are given. E_{max} is only introduced to allow the computation of percentage changes of the state of charge. Furthermore, it is assumed that c_E is almost independent of the current state of charge, only equation 6 is considered, i.e., a c_E-value, which offends this condition for an edge e, at time t and a state of charge SOC, is adapted in the following way:

$$\text{if } E_{max} \cdot SOC - \tilde{c}_E(e,t,SOC) > 1, \text{ then} \tag{10}$$
$$c_E(e,t,SOC) := SOC - 1 \tag{11}$$
$$\text{if } E_{max} \cdot SOC - \tilde{c}_E(e,t,SOC) < 0, \text{ then} \tag{12}$$
$$c_E(e,t,SOC) := SOC \tag{13}$$
$$\text{else} \tag{14}$$
$$c_E(e,t,SOC) := \frac{\tilde{c}_E(e,t,SOC)}{E_{max}} \tag{15}$$

Now, it is possible to compute the energy consumption costs of path $P := (a, c, d)$, when starting at time $t_S = 0$. The time at which node c is reached on P is denoted with t_c. The corresponding state of charge is given by SOC_c:

$$t_c := t_S + c_T((a,c),t_S) = 0 + 1 = 1 \tag{16}$$
$$SOC_c := SOC_S - c_E((a,c),t_S,SOC_S) \tag{17}$$
$$= 1.00 - \frac{4}{E_{max}} = 0.6 \tag{18}$$

Then, the state of charge $SOC(P, t_S, SOC_S)$ after driving along path P when starting at time t_S can be computed as follows:

$$SOC(P,t_S,SOC_S) = SOC_S - c_E(P,t_S,SOC_S) \tag{19}$$
$$= SOC_c - c_E((c,d),t_c,SOC_c) \tag{20}$$
$$= 1.0 - 0.4 - 0.4 = 0.2 \tag{21}$$

P is an energy-secure path according to definition 1. This would not be the case if SOC_{min} would be set to more than 0.2.

An important aspect of this first example is that the energy consumption costs could be reduced if the car waits at the beginning until $t'_S = 1$:

$$t'_c := t'_S + c_T((a,c),t'_S) \; = 1 + 1 = 2 \tag{22}$$
$$SOC'_c := SOC_S - c_E((a,c),t'_S,SOC_S) = 0.6 \tag{23}$$

Then, for the resulting state of charge it holds:

$$SOC(P,t'_S,SOC_S) = SOC_S - c_E(P,t_S,SOC_S) \tag{24}$$
$$= SOC_S - c_E((a,c),t'_S,SOC_S) \tag{25}$$
$$- c_E((c,d),t'_c,SOC'_c) \tag{26}$$
$$= 1 - 0.4 - 0.1 = 0.5 \tag{27}$$

The reason for this is that the energy consumption costs for the edge (c, d) are significantly lower if one starts at node c at $t > 1$. However, proposing waiting times seems not to be relevant for the practical application of navigation-services and it is in a realistic scenario counter-productive for minimizing the time costs. Thus, waiting times are not considered in the remaining part of the paper.

B. Multicriteria Optimization

The notion of optimality for multicriteria minimization problems leads to several issues. For instance, considering the example graph shown in Fig. 2. If one compares the a-d-paths $P_1 = (a, d)$ and $P_2 = (a, b, c, d)$. This leads for $t_S = 0$ and $SOC_S = 1.0$ to costs

$$c(P_1,t_S,SOC_S) := \begin{pmatrix} c_T(P_1,t_S) \\ c_E(P_1,t_S,SOC_S) \end{pmatrix} = \begin{pmatrix} 5 \\ 0.5 \end{pmatrix} \tag{28}$$
$$c(P_2,t_S,SOC_S) := \begin{pmatrix} c_T(P_2,t_S) \\ c_E(P_2,t_S,SOC_S) \end{pmatrix} = \begin{pmatrix} 3 \\ 0.6 \end{pmatrix} \tag{29}$$

Deciding which path is "better" is not trivial, since P_2 leads to lower time costs, P_1 to lower energy consumption costs. For such situations, the concept of Pareto optimality was introduced. Here, one possible formulation of Pareto

optimality for cheapest path problems in time-dependent networks is stated:

Definition 2. *Let a finite and directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$, a starting node s, a destination node d, a starting time t_S and Z time-dependent cost functions $c_1, c_2, ..., c_Z$ be given. Then, for a starting time t_S, a s-d-path P^* is called* ***Pareto optimal*** *if and only if there is no other s-d-path $\bar{P}$ with*

$$c(\bar{P}, t_S) = \begin{pmatrix} c_1(\bar{P}, t_S) \\ \vdots \\ c_Z(\bar{P}, t_S) \end{pmatrix} \lneq c(P^*, t_S) = \begin{pmatrix} c_1(P^*, t_S) \\ \vdots \\ c_Z(P^*, t_S) \end{pmatrix}. \tag{30}$$

Hereby, for two vectors $c^* := (c_1^*, ..., c_Z^*)$ and $\bar{c} := (\bar{c}_1, ..., \bar{c}_Z) \in \mathbb{R}^Z$, it is $\bar{c} \lneq c^*$ if and only if the following two conditions hold:

$$\bar{c}_i \leq c_i^* \quad \forall i \ \in \{1, ..., Z\} \tag{31}$$

and there is at least one index $j \in \{1, ..., Z\}$ with

$$\bar{c}_j < c_j^*. \tag{32}$$

If $c(\bar{P}, t_S) \lneq c(P^*, t_S)$, then it is written that $\bar{P}$ ***dominates*** *P^* (for starting time t_S).*

The problem of finding all Pareto optimal paths on a directed graph $\overrightarrow{G}$ for a starting time t_S, a starting node s, a destination node d and cost functions $c_1, ..., c_Z$ is then stated as:

$$\min_{\lneq} c(P, t_S) = \begin{pmatrix} c_1(P, t_S) \\ \vdots \\ c_Z(P, t_S) \end{pmatrix} \tag{33}$$

$$\text{subject to: } P \text{ is a } s\text{-}d\text{-path on } \overrightarrow{G}. \tag{34}$$

Remembering the paths P_1 and P_2 from before, both paths are Pareto optimal for $t_S = 0$ due to definition 2 (under the assumption of a fixed SOC_S). In general, the number of Pareto optimal paths can be very high. Actually, even for only two cost functions, it can increase exponentially with the number of nodes in V (see theorem 9.3 in [5]). This is one of the main reasons leading to an extremely high computational effort when generating the whole set of Pareto optimal paths. To avoid unbearable computation times, several approaches were developed to reduce multicriteria optimization problems to single-criteria optimization problems in such a way that solving the new problem still leads to Pareto optimal or at least to "good" solutions. Besides the weighted sum scalarization, where a weighted sum of cost functions is considered as new objective function, the so-called ϵ-constraint method (see section 4.1 in [5]) is one of the most common ideas: Instead of trying to optimize all cost functions $c_1, ..., c_Z$ simultaneously, only the cost function c_1 is chosen as objective function. The other cost functions $c_2, ..., c_Z$ are allowed to obtain any value, as long as certain upper bounds $\epsilon_2, ..., \epsilon_Z$ are respected:

$$\min_{\lneq} c_1(P, t_S) \tag{35}$$

$$\text{subject to: } P \text{ is a } s\text{-}d\text{-path on } \overrightarrow{G} \tag{36}$$

$$c_i(P, t_S) \leq \epsilon_i \quad \forall i \ \in \{2, ..., Z\} \tag{37}$$

For the case of two cost functions and a path $P = [v_1, ..., v_Z]$, the ϵ-constraint problem mirrors perfectly the idea of energy-secure time-optimal paths if the following definitions are made:

$$c_1(P, t_S) := c_T(P, t_S) \tag{38}$$

$$c_2(P, t_S) := \max\{c_E(P_{1:i}, t_S, SOC_S) \mid i \leq n\} \tag{39}$$

$$\epsilon_2 := SOC_S - SOC_{min} \tag{40}$$

Note that c_2 depends not only on time, but also on the state of charge.

C. Dijkstra's Limits

In this paper, the optimization is done w.r.t. c_T. Consequently, to avoid confusion, it is not spoken of "shortest" path problems, but of "cheapest" path problems. One of the most common algorithms for solving cheapest path problems is Dijkstra's algorithm [3]. In the following, modified versions of this algorithm are applied in order to find for a starting time t_S, a graph $\overrightarrow{G}$, a starting node s, an initial state of charge SOC_S, a minimum state of charge SOC_{min} and a destination node d a time-optimal and energy-secure s-d-path P^* on $\overrightarrow{G}$. Clearly, some kind of modification is necessary, because Dijkstra's algorithm in its basic version can neither handle general time-dependent networks, nor cost functions that assume negative values, nor several cost functions at the same time. Solution methods meeting any of these conditions have already been developed. For example, an approach to use Dijkstra's algorithm also for negative costs especially in the context of electric vehicles is shown in [10]. The problem of time-dependency is intensively discussed in [11], [8], [17] and for multicriteria cheapest path problems an extension of Dijkstra's algorithm is stated in [15]. More recent work even considers combinations of the above mentioned problems: In [14] and [4], algorithms for solving multicriteria cheapest path problems in time-dependent networks are shown. In [9] additionally the topic of negative costs is covered. However, any algorithm proposed in these papers which is capable of handling several criteria generates the whole set of Pareto optimal paths and thus suffer from very high computation times. It has already been stated that in this paper one is interested in just finding one time-optimal solution. Due to this, different requirements are of interest, which leads to the fact that algorithm 1 from [15] seems to be most suitable. In its basic version, this algorithm is intended for multicriteria cheapest path problems. It returns the set of all Pareto optimal paths, but it cannot handle negative costs. Also time-dependency has not been considered in [15]. The huge advantage of this algorithm is that it is a label-setting algorithm. In contrast to label-correcting algorithms, as for example in [9], algorithms of this type can typically be modified in

such a way that they stop as soon as one (Pareto) optimal path is computed, a fact that can reduce computation times drastically. Certainly, label-correcting algorithms are usually used for handling negative costs as in most cases label-setting algorithms cannot do this. For the proposed algorithm, this will be no problem, as not a general multicriteria cheapest path problem will be considered, but an ϵ-constraint problem, where the optimization is done according to the nonnegative cost function c_T. When proving the correctness of algorithm A, it can be observed that neither the negativity of c_E, nor its dependency on the state of charge have any impact within the proof.

III. Computation of Energy-secure Time-optimal Paths

In this section, an algorithm (denoted as algorithm A) for finding optimal charging strategies is introduced. It uses the following definition:

Definition 3. *For two Z-dimensional vectors $c^* := (c_1^*, c_2^*, ..., c_Z^*)$, $\bar{c} := (\bar{c}_1, \bar{c}_2, ..., \bar{c}_Z) \in \mathbb{R}^Z$, the vector c^* is denoted as **lexicographically** smaller than $\bar{c}$ if $c^* = \bar{c}$ or if $c_j^* < \bar{c}_j$ with $j := \min\{i : c_i^* \neq \bar{c}_i,\ i \in \{1, ..., Z\}\}$. Alternatively, one can write $c^* \leq_{lex} \bar{c}$.*

Furthermore, the notion of labels is used. Here, a label $L := (c_T, c_E, v^{pre}, n^{pre}, v^{cur}, n^{cur})$ is a 6-tuple, i.e., an ordered set of size six. Each label belongs to a specific node, storing additional information about how the node is reached during the route finding process: The first entry contains the cumulated time costs of the considered node on the current path, the second entry the cumulated energy consumption costs. The third entry is the preceding node on the current path. The fourth entry is the index of the label of the preceding node. As different paths, which leads to the same node, may lead to varying arrival times or a different state of charge at this node, several labels may belong to the same node. Consequently, an index is necessary to distinguish between these labels. The fifth entry is the current node, i.e., the node the label belongs to, and the last entry is the corresponding index. Note that a label belongs to a node, but it also allows the reconstruction of a path leading to this node. Actually, by referring to its preceding label, each label encodes a path. In the basic version of Dijkstra's algorithm, labels are assigned to nodes, too. However, these labels only contain the costs for getting to this node and the information about the preceding node.

The proceeding of algorithm A is comparable to Dijkstra's algorithm. In the following, it is explained for the case of the example stated in Fig. 2. The starting node is a and the destination node is d. Similar to Dijkstra's algorithm, there is a set of temporary labels $\mathcal{L}_{temp}$ and a set of permanent labels $\mathcal{L}_{perm}$. Note that, in contrast to Dijkstra's algorithm, algorithm A cannot delete any labels, even labels of $\mathcal{L}_{temp}$. The development of these sets during the initialization step and all five iterations of the while-loop can be found in Tab. I. Labels are ordered lexicographically according to the costs which are associated with them. For a label $L = (c_T^{cur}, c_E^{cur}, v^{pre}, n^{pre}, v^{cur}, n^{cur})$, the corresponding costs $c(L, t_S, SOC_S)$ are defined as the cumulative costs

$$c(L, t_S, SOC_S) := (c_T^{cur}, c_E^{cur}) \tag{41}$$

Consequently, the labels in $\mathcal{L}_{temp}$ are primarily ordered according to their first entry, i.e., according to the cumulated time costs which they encode. If several labels have the same first entry, then these labels are ordered according to their second entry.

Iteration	$\mathcal{L}_{temp}$	$\mathcal{L}_{perm}$
	$L_a^1 := (0, 0\%, \emptyset, 0, a, 1)$	
It. 1	$L_b^1 := (1, 20\%, a, 1, b, 1)$, $L_c^1 := (1, 40\%, a, 1, c, 1)$ $L_d^1 := (5, 50\%, a, 1, d, 1)$	L_a^1
It. 2	L_c^1 $L_c^2 := (2, 50\%, b, 1, c, 2)$ L_d^1	L_a^1, L_b^1
It. 3	L_c^2 $L_d^2 := (4, 80\%, c, 1, d, 1)$ L_d^1	L_a^1, L_b^1, L_c^1
It. 4	$L_d^3 := (3, 60\%, c, 2, d, 3)$ L_d^2 L_d^1	L_a^1, L_b^1, L_c^1, L_c^2
It. 5	L_d^2 L_d^1	L_a^1, L_b^1, L_c^1, L_c^2, L_d^3

TABLE I

Proceeding of algorithm A for the example from Fig. 2

During the initialization the label $L_a^1 := (0, 0\%, \emptyset, 0, a, 1)$ belonging to the starting node a is generated and added to the temporal labels $\mathcal{L}_{temp}$. The entries of L_a^1 result directly. As there is no preceding node, the third and the fourth entry of L_a^1 are simply set to $\emptyset$ and 0. Before the first iteration of the while-loop, since the termination criteria in line 1 of algorithm A cannot be fulfilled, the only element of $\mathcal{L}_{temp}$ is selected, i.e., in line 2 of algorithm A it is $L^{cur} := L_a^1$. The label is removed from the set of temporal labels and added to the permanent labels $\mathcal{L}_{perm}$. Then, in lines 4 to 7, new labels are created for all neighbouring nodes of a. Note that n^{new} in line 7 of algorithm A is simply defined by increasing the number of existing labels for node v^{new} by 1. If there is no existing label, then $n^{new} := 1$. For the considered example, there are three labels generated, one for node b, c and d. As shown in Tab. I, these labels are added to the set $\mathcal{L}_{temp}$ as all of them fulfill the energy-security condition in line 8. At the start of the second iteration label L_b^1 is selected as the lexicographically smallest temporary label and added to $\mathcal{L}_{perm}$. With L_c^2 only one new label is created. Note that at this time, two different labels belonging to node c exist in $\mathcal{L}_{temp}$. The algorithm proceeds analogously three further iterations until it terminates after making the label L_d^3 permanent, which is the first label in $\mathcal{L}_{perm}$ that belongs to the destination node d. If, after the termination of algorithm A, there is a permanent label which belongs to the destination node, then a solution, i.e., an energy-secure path from the start to the destination, has been found. The

Algorithm A: Multicriteria Cheapest Path Search in Time-dependent Network with Negative Edge Costs

Input: A directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$, a starting node s, a starting time $t_S := 0$, a destination node d, two cost functions c_T and c_E as described above, a starting state of charge SOC_S and minimal allowed state of charge SOC_{min}

Initialization: Create label $L = (0, 0, \emptyset, 0, s, 1)$ for node s and define $\mathcal{L}_{temp} := \{L\}$

1 While $\mathcal{L}_{temp} \neq \emptyset$ and no label belonging to the destination node was added to $\mathcal{L}_{perm}$, do:
2 Let $L_{cur} = (c_T^{cur}, c_E^{cur}, v^{pre}, n^{pre}, v^{cur}, n^{cur})$ be the lexicographically smallest label in $\mathcal{L}_{temp}$.
3 Remove L^{cur} from $\mathcal{L}_{temp}$ and add it to $\mathcal{L}_{perm}$.
4 For all $v^{new} \in V$ such that $e := (v^{cur}, v^{new}) \in \overrightarrow{E}$ do:
5 Compute $c_T^{new} := c_T^{cur} + c_T(e, t_S + c_T)$
6 Compute $c_E^{new} := c_E^{cur} + c_E(e, t_S + c_T, SOC_S - c_E^{cur})$
7 Create $L^{new} := (c_T^{new}, c_E^{new}, v^{cur}, n^{cur}, v^{new}, n^{new})$
8 If $SOC_S - c_E^{new} \geq SOC_{min}$, then:
9 add L^{new} to $\mathcal{L}_{temp}$
10 End if.
11 End for.
12 End while.
13 If possible, return a label $\bar{L} \in \mathcal{L}_{perm}$ that belongs to node d,
14 otherwise return "No feasible solution found".

reconstruction of the computed path can be done similar to Dijkstra's algorithm, by starting with the only permanent label belonging to the destination node and successively adding predecessors until a label is reached, which belongs to the starting node. Continuing the example from above, one has to start with L_d^3. Its preceding label is encoded in its entries three and four, namely the label L_c^2. The predecessor of label L_c^2 is L_b^1, the predecessor of L_b^1 is L_a^1. Hence, the computed path is (a, b, c, d). The costs of the path are according to equation 41 given by:

$$c(L_d^1, 0, 100\%) = \begin{pmatrix} 3 \\ 60\% \end{pmatrix}. \tag{42}$$

A. Proving Optimality of Algorithm A

Theorem 1. *Let a finite and directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$, a starting node s, a destination node d, two time-dependent cost functions c_T and c_E (as described above), a starting state of charge SOC_S and minimal allowed state of charge SOC_{min} be given. Furthermore, let there be no cycle on $\overrightarrow{G}$ that leads to time costs of zero and let at least one optimal solution for the problem of finding a fastest and energy-secure path from s to d on $\overrightarrow{G}$ under the given conditions exist. Then, algorithm A terminates with finding a label which encodes a time-optimal and energy-secure s-d-path.*

The condition that all cycles on $\overrightarrow{G}$ have to lead to positive time-costs is only necessary to ensure that algorithm A does not end up within an infinity-loop. This guarantees, together with the finiteness of $\overrightarrow{G}$ and the existence of at least one solution, that algorithm A terminates with finding a label which belongs to d. According to this, the following proof will only show that the computed label encodes a time-optimal path.

Proof Let $P^* = [v_1^* = s, ..., v_K^* = d]$ denote a time-optimal and energy-secure path for the problem described in theorem 1 and let L_k^* denote the label corresponding to the k-th node v_k^*. Furthermore, let $\bar{L}$ be the label computed by algorithm A and let $\bar{P} = (\bar{v}_1 = s, ..., \bar{v}_Q = d)$ be the corresponding path, i.e., at the end of algorithm A, the label $\bar{L}$ is added to the set of permanent labels $\mathcal{L}_{perm}$. The energy-security of $\bar{P}$ is trivially ensured by the condition of line 8 in algorithm A. Hence, it is now assumed that $\bar{P}$ is not time-optimal, i.e.

$$c_T(\bar{P}, t_S) > c_T(P^*, t_S). \tag{43}$$

As $c_T(e, t) \geq 0 \; \forall e \in \overrightarrow{E}$ and $\forall t$, it holds that the time costs

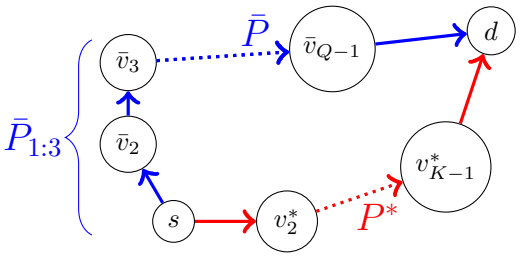


Fig. 3. Proving optimality of algorithm A

of all subpaths $P_{1:k}^*$ with $k \in \{1, 2, ..., K\}$ of path P^* are at most as high as $c_T(P^*, t_S)$ and thus these costs are smaller than the time costs for the computed path $c_T(\bar{P}, t_S)$:

$$c_T(L_k^*, t_S) = c_T(P_{1:k}^*, t_S) < c_T(\bar{P}, t_S) = c_T(\bar{L}, t_S). \tag{44}$$

This implies that all labels L_k^* encodes lexicographically lower costs than $\bar{L}$:

$$c(L_k^*, t_S, SOC_S) = \tag{45}$$
$$(c_T(P_{1:k}^*, t_S), c_E(P_{1:k}^*, t_S, SOC_S)) \leq_{lex} \tag{46}$$
$$(c_T(\bar{P}, t_S), c_E(\bar{P}, t_S, SOC_S)) = c(\bar{L}, t_S, SOC_S). \tag{47}$$

Note that each label $\bar{L}_q$ have to fulfill the condition in line 8 of algorithm A, since $\bar{P}$ is energy-secure and thus, due to definition 1, all its subpaths are energy-secure, too. Using mathematical induction, it is now shown that all labels L_k^*

are set permanent before the label $\bar{L}$, which encodes the path $\bar{P}$. This implies also that L_K^* is added to $\mathcal{L}_{perm}$ before $\bar{L}$ and hence the path P^* is found before $\bar{P}$. As algorithm A terminates in line 1 if a permanent label for the destination node is set, this implies that algorithm A terminates before the label $\bar{L}$ is set permanent, which is a contradiction to the assumption that algorithm A ends with finding $\bar{L}$.

Start of induction: As P^* is a s-d-paths on $\overrightarrow{G}$, the first label for this path is given by

$$L_1^* = (0, 0, \emptyset, 0, s, 1). \tag{48}$$

This label is set permanent during the first iteration of the while-loop in algorithm A and thus is it is added to $\mathcal{L}_{perm}$ before $\bar{L}$.

Inductive step: Let label L_k^* with $k < K$ be already added to $\mathcal{L}_{perm}$ and $\bar{L} \notin \mathcal{L}_{perm}$. As P^* is a path on $\overrightarrow{G}$, it is $(v_k^*, v_{k+1}^*) \in \overrightarrow{E}$. Thus, at that time when L_k^* was added to $\mathcal{L}_{perm}$, the label L_{k+1}^* is created in line 7. Since P^* and all its subpaths are energy-secure, the label L_{k+1}^* is added to $\mathcal{L}_{temp}$ in the same iteration of the while-loop in which L_k^* was added to $\mathcal{L}_{perm}$. Hence, $\bar{L}$ cannot be in $\mathcal{L}_{perm}$ before L_{k+1}^* is added to $\mathcal{L}_{temp}$. Thus, according to the term in 45 - 47, the label L_{j+1}^* is added earlier to $\mathcal{L}_{perm}$ than $\bar{L}$. □

A big advantage of algorithm A is that it returns an energy-secure and time-optimal path even if under very weak assumptions: Nonnegative time costs are, from a practical perspective, no restriction. Furthermore, no FIFO-property for c_T [18], nor any conditions concerning the energy consumption costs c_E are made. However, this leads to drawbacks for the computational speed:

Lemma 1. *Let a finite and directed graph $\overrightarrow{G} = (V, \overrightarrow{E})$, a starting node s, a destination d, two time-dependent cost functions c_T and c_E (as described above), a starting state of charge SOC_S and minimal allowed state of charge SOC_{min} be given. Moreover, let a node $\bar{v}$, a finite energy-secure s-$\bar{v}$-paths $\bar{P} = [\bar{v}_1 = s, \bar{v}_2, ..., \bar{v}_Q = \bar{v}]$ on $\overrightarrow{G}$, as well as a finite, energy-secure and time-optimal s-d-path P^* on $\overrightarrow{G}$ be given with*

$$c_T(\bar{P}, t_S) < c_T(P^*, t_S). \tag{49}$$

Then, if algorithm A is used to compute a time-optimal and energy-secure s-d-path on $\overrightarrow{G}$, no label encoding a time-optimal and energy-secure path is added to $\mathcal{L}_{perm}$ before all labels encoding subpaths of $\bar{P}$ are added to $\mathcal{L}_{perm}$.

Proof The proof of lemma 1 can be done via mathematical induction, analogously to the proof of theorem 1. The only difference is that now one has to show that all labels encoding subpaths of $\bar{P}$ are found before any label encoding a time-optimal path.

Lemma 1 states that algorithm A computes any possible path which is energy-secure and leads to time-costs lower than the optimal time-costs for a s-d-path on $\overrightarrow{G}$. The reason for this is that, in contrast to (for example) Dijkstra's algorithm, no label is deleted because it encodes a "bad" path. Considering Tab. I, label L_c^2 leads for getting from node a to node c to higher time- and consumption-costs than L_c^1, i.e., it is dominated by another label. Nevertheless, L_c^2 is added to $\mathcal{L}_{temp}$ and finally even to $\mathcal{L}_{perm}$. Compared to existing algorithms for solving multicriteria cheapest path problems, which typically deletes dominated labels [15] and still suffer from an exponential growth of computation time [5], algorithm A would behave even worse until it terminates. A possible way to delete at least all dominated labels is provided by algorithm B: It is constructed as a modification of algorithm A. The original for-loop is extended by adding additional conditions which allows to delete all dominated labels. This reduces the number of computed paths and thus computation time. Algorithm B is very similar to algorithm 1 in [15]. The only differences are the energy-security condition stated in line 9 of algorithm B and that algorithm B terminates as soon as a label belonging to the destination is added to $\mathcal{L}_{temp}$. Due to this also their computational behavior is comparable. Analogously to Tab. I, Tab. II describes the proceeding of algorithm B for the small example graph of Fig. 2. Obviously, for the considered test case the number of iterations and especially the number of created labels is, compared to Tab. I, reduced. The condition in line 7 of algorithm B ensures that in iteration 3, when label L_b^1 is added to $\mathcal{L}_{perm}$, label $L_c^2 = (2, 50\%, b, 1, c, 2)$ is, in contrast to algorithm A, not added to $\mathcal{L}_{temp}$, since it is dominated by L_c^1. However, L_c^2 is part of the time-optimal path $[a, b, c, d]$.

Iteration	$\mathcal{L}_{temp}$	$\mathcal{L}_{perm}$
	$L_a^1 := (0, 0\%, \emptyset, 0, a, 1)$	
It. 1	$L_b^1 := (1, 20\%, a, 1, b, 1)$, $L_c^1 := (1, 40\%, a, 1, c, 1)$ $L_d^1 := (5, 50\%, a, 1, d, 1)$	L_a^1
It. 2	L_c^1 L_d^1	L_a^1, L_b^1
It. 3	$L_d^2 := (4, 80\%, c, 1, d, 1)$ L_d^1	L_a^1, L_b^1, L_c^1
It. 4	L_d^1	L_a^1, L_b^1, L_c^1, L_c^2, L_d^2

TABLE II
PROCEEDING OF ALGORITHM B FOR THE EXAMPLE FROM FIG. 2

Hence, algorithm B cannot guarantee time-optimal solutions. The critical issue is that in multicriteria time-dependent networks Bellman's optimality principle [1] does not necessarily hold [9]. Bellman optimality principle (or the Bellman equation) forms typically the fundament of dynamic programming approaches like Dijkstra's algorithm. Solutions to problems sufficing this principle have the property that any sub-solution is optimal for the corresponding sub-problem. Applied to the graph of Fig. 2, if Bellman's optimality principle holds, then any subpath of the time-optimal path $[a, b, c, d]$ should be a time-optimal solution for the corresponding start and destination node, too. Certainly, the path $[a, b, c]$ is a subpath of the time-optimal path, but

Algorithm B: Modification of Algorithm A for Accelerated Computation

...

4 For all $v^{new} \in V$ such that $e := (v^{cur}, v^{new}) \in \overrightarrow{E}$ do:

5 Compute $c_T^{new} := c_T^{cur} + c_T(e, t_S + c_T)$

6 Compute $c_E^{new} := c_E^{cur} + c_E(e, t_S + c_T, SOC_S - c_E^{cur})$

7 Create $L^{new} := (c_T^{new}, c_E^{new}, v^{cur}, n^{cur}, v^{new}, n^{new})$

8 If L^{new} is not dominated by any other label in $\mathcal{L}_{temp}$ or $\mathcal{L}_{perm}$ that belongs to v^{cur}

9 and if $SOC_S - c_E^{new} \geq SOC_{min}$, then:

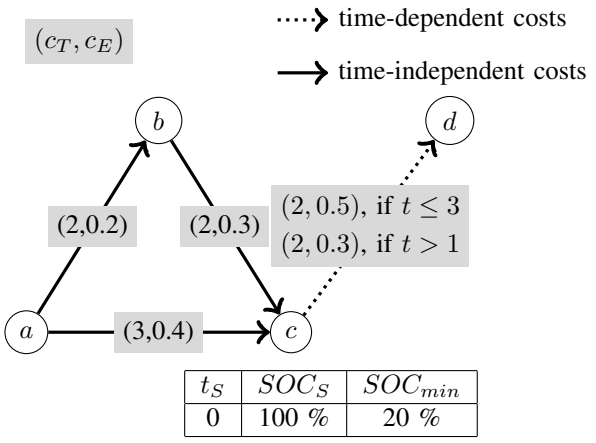
10 add L^{new} to $\mathcal{L}_{temp}$ and delete all labels belonging to v^{cur} in $\mathcal{L}_{temp}$ that are dominated by L_{new}.

11 End if.

12 End for.

...

not a solution to the problem of finding an energy-secure and time-optimal path from a to c when starting at $t_S = 0$ with an initial state of charge $SOC_S = 100\%$. The possible existence of dominated subpaths as parts of optimal paths is a huge drawback. It prompts for the case of charging strategy optimization that dynamic programming approaches in general cannot reduce their search space and simultaneously ensure to find an time-optimal and energy-secure solution. Note that algorithm B not only looses optimality: When trying to compute an energy-secure and time-optimal a-d-path, Fig. 4 illustrates a small example, where algorithm B does not lead to any solution at all, even though a feasible solution exists. Similar to the problem stated in Fig. 2, algorithm B does not



t_S	SOC_S	SOC_{min}
0	100 %	20 %

Fig. 4. Limits of algorithm B

add label $L_c^2 = (4, 0.5, b, 1, c, 2)$ to $\mathcal{L}_{temp}$. Unfortunately, this label is necessary to construct the only energy-secure a-d-path $[a, b, c, d]$. The label $L_d^1 = (5, 90\%, c, 1, d, 1)$ is created during iteration 3, but it encodes no energy-secure path as SOC_{min} is equal to 20% and the path $[a, c, d]$, which is encoded by L_d^1, leads to a state of charge of

$$SOC_S - c_E([a, c, d], 0, 100\%) = 100\% - 90\% = 10\%. \quad (50)$$

Consequently, this label does not fulfill the energy-security condition in line 9 of algorithm B and it is not added to $\mathcal{L}_{temp}$. Note that algorithm B would return time-optimal solutions if Bellman's optimality principle holds. This could

Iteration	$\mathcal{L}_{temp}$	$\mathcal{L}_{perm}$
	$L_a^1 := (0, 0\%, \emptyset, 0, a, 1)$	
It. 1	$L_b^1 := (2, 20\%, a, 1, b, 1)$, $L_c^1 := (3, 40\%, a, 1, c, 1)$	L_a^1
It. 2	L_c^1	L_a^1, L_b^1
It. 3		L_a^1, L_b^1, L_c^1

TABLE III

PROCEEDING OF ALGORITHM B FOR THE EXAMPLE FROM FIG. 4

be proven similar to theorem 1. One has to additionally show that all labels which are deleted due to the new condition in line 8 or the new instruction in line 10 of algorithm B are not preceding labels of a label which encodes a time-optimal and energy-secure solution. But this follows due to the fact that all preceding labels of labels which encodes a time-optimal and energy-secure solution have to encode a non-dominated path. From a practical perspective, it seems reasonable to ignore dominated sub-strategies, although first computational results indicate that even in a realistic scenario, i.e., with realistic cost functions, Bellman's optimality principle not necessarily holds.

IV. MODELING CHARGING PROCESSES

Now, charging stations are modeled as a part of a directed graph. To do this, a setting is assumed as visualized in the upper part of Fig. 5: A charging station is located along the road segment which is represented by the edge (a, b). The idea is to extend the graph in such a way that paths may also "visit" the charging station. In addition, varying charging durations should be possible. One approach to do this can be found in the lower part of Fig. 5. Here, charging durations of 5, 10, 15, ..., 60 minutes are modeled. Starting from node a, one either can get directly to node b, or one drives to the charging station denoted by nodes e and l_d with $d \in \{5, 10, 15, ..., 60\}$. The time costs for the edges (a, e) and (l_d, b) have to represent the duration for driving from node a to the charging station or from the charging station to node b, respectively. The same holds for the energy consumption costs. Important in this context is that on edge (a, e) an electric vehicle always decelerates its speed down to zero and on edge (l_d, b) it accelerates starting from a speed of

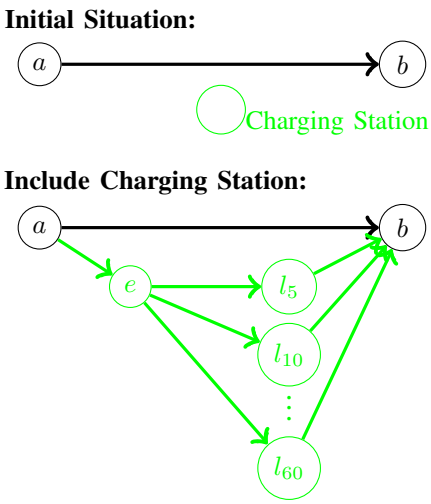


Fig. 5. Modeling of charging stations

zero. However, the most interesting part is the cost-modeling for the edges (e, l_d). These edges are intended to represent the charging process itself, i.e., the edge (e, l_d) represents a charging process of duration d. For the current considerations d is given in minutes, i.e., the edge (e, l_{20}) corresponds to a charging duration of 20 minutes. Consequently, assuming that one can start charging immediately, independent of the arrival time t, it is $c_T((e, l_d), t) := d$. Note that one could also add additional time for paying the charged energy or getting off and into the car. To define the energy "consumption" costs when charging, information concerning the relation between charging duration and charged energy is necessary. Here, it is assumed that a function $S : \mathbb{R}_{\geq 0} \longrightarrow [0, 1]$ is given, which returns for a given charging duration d the resulting state of charge if one assumes that the initial state of charge, i.e., the state of charge at the beginning of the charging duration, is equal to zero. An example of how such a function may look like, can be found in Fig. 6. It has

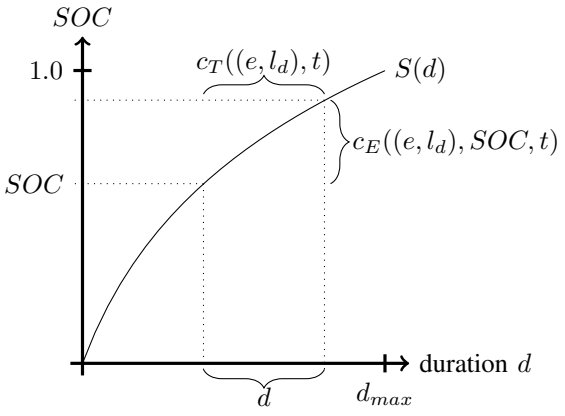


Fig. 6. Relation between charging duration and state of charge

already been mentioned that typically less energy can be charged during a certain time interval if the battery is almost fully charged. Hence, S is concave. Furthermore, as a longer charging durations obviously lead to a higher state of charge, it follows that S is monotonically increasing until the battery is fully charged. The time which is needed to fully recharge a battery with an initial state of charge of 0 is denoted with d_{max}. Moreover, $S(d) \equiv 1$ for $d \geq d_{max}$. Consequently, the function S can be inverted on the interval $[0, d_{max}]$, i.e., $S^{-1} : [0, 1] \longrightarrow [0, d_{max}]$. Note that for the nodes l_d, a value $d > d_{max}$ is not necessary, since even for an initial state of charge of 0 the battery would be already fully charged after a charging duration of d. Due to this, the highest possible charging duration should be set equal to d_{max}. By assuming that the charging duration for charging from a value SOC_1 up to a value SOC_2, with $SOC_1 \leq SOC_2$, is always equal to the difference between $S^{-1}(SOC_2)$ and $S^{-1}(SOC_1)$, it can be concluded for any charging duration $d \in \{5, 10, ..., 60\}$:

$$c_E((e, l_d), t, SOC) := -S(d + S^{-1}(SOC)) + SOC \quad (51)$$

Note that such edge costs are negative, but algorithm A still finds a time-optimal and energy-secure solution, since all conditions of theorem 1 are fulfilled.

V. Application Possibilities and Limits

Algorithms 1 and 2 were originally developed for a very restricted setting: The starting time and the starting state of charge are variable, but given. Certainly, the starting point, the destination and even the route and potential charging stations are fixed. Within the project (see section "acknowledgments"), in which this research takes place, a route starting in Munich and leading via the German autobahn A9 to Leipzig is analyzed. Along this more than 500 kilometers long road corridor altogether 7 fast charging stations are placed. Additionally, one fast charging station is located at the beginning and one at the end of the route. The possibility to charge fast, i.e., one needs roughly 25 minutes to charge an empty battery up to a state of charge of 80%, is in this context very important. Conventional charging would lead to charging durations of several hours, which is inappropriate for recharging batteries during a trip. First computational results for this limited setting shows that algorithm B is sufficiently fast, whereas algorithm A even in this small setting is quite slow. Thus, algorithm A is mainly useful for an ex post determination of the quality of solutions computed by algorithm B and hence to estimate the applicability of algorithm B.

The restriction to one specific route does only allow to avoid traffic congestions temporarily, but not to circumvent them spatially. Lots of optimization potential is lost. However, a generalization to arbitrary road networks probably leads to unacceptably high computation time, even for algorithm B. The critical issue is that the problem of finding cheapest paths can (at least up to the current state of art) neither be solved in polynomial time for general time-dependent networks [17], nor for the multicriteria case [5]. Nevertheless, recently so-called preprocessing methods were introduced, for example in [19] or [7]. Such approaches are used to compute apriori, i.e., before the first route

search is carried out, additional information for a given graph. Afterwards, this additional information is used by routing algorithms to speed up their computation times. This allows to find optimal solutions even for time-dependent or multicriteria routing problems in moderate time (chapters 5 and 6 in [2]). Note that most of these procedures need the computation of many optimal paths for the preprocessing. Both, time-dependency, as well as the existence of several criteria increase the computational effort. The existence of negative costs and, remembering lemma 1, especially the fact that Bellman's optimality principle does not hold may increase computations times further. Even though fast computation times for the preprocessing methods are not necessary (since the preprocessing has to be done only once), the corresponding computations at least have to be executable in reasonable time. All in all, using preprocessing methods seems inevitable if one wants to apply the proposed algorithms to general road networks, but a detailed analysis of potential preprocessing methods has to be carried out in order to understand which methods can in general be applied and which of these methods may lead to substantial accelerations.

VI. Summary

Goal of the paper was the development of a point-to-point cheapest path problem and a corresponding optimization algorithm in order to generate optimal charging strategies for electric vehicles on long trips. At the beginning, the focus was set on analyzing properties of the cost functions c_T and c_E, which describe the time consumption and the energy consumption, respectively. To mirror reality adequately, it was explained why both cost functions have to be considered simultaneously, why both functions have to be time-dependent and why c_E can assume negative values. Each of these three properties causes difficulties for cheapest path searches. Nevertheless, an algorithm has been proposed which is capable of handling all of these issues. It has been proven that this algorithm finds a time-optimal solution which additionally guarantees an energy-secure arrival at the destination and it was argued that this algorithm leads to very high computation times. A modification of this algorithm, leading in general to non-optimal solutions, was introduced in order to reduce computational effort. The possibility of charging during the trip was included in the graph-based routing model and a small discussion concerning potential applications of the proposed algorithms was given.

VII. Acknowledgments

The authors want to thank the German Bundesministerium für Verkehr, Bau und Stadtentwicklung, which funds the project "DC-Ladestation am Olympiapark". This project, in which the described research took place, is a sub-project of the "Schaufenster Bayern-Sachsen ELEKTROMOBILITÄT VERBINDET".

References

[1] Richard Bellman. The theory of dynamic programming. *Bull. Amer. Math. Soc.*, pages 503–515, 1954.

[2] Delling and Daniel. *Engineering and Augmenting Route Planning Algorithms*. Universität Fridericiana zu Karlsruhe, 2009.

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

[4] Yann Disser, Matthias Müller-Hannemann, and Mathias Schnee. Multi-criteria Shortest Paths in Time-Dependent Train Networks. In Catherine C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 347–361. Springer Berlin Heidelberg, 2008.

[5] Matthias Ehrgott. *Multicriteria optimization*. Springer, Berlin and New York, 2 edition, 2005.

[6] Eva Ericsson. Variability in exhaust emission and fuel consumption in urban driving. *Urban Transport Systems. Proceedings from the 2nd KFB Research Conference in Lund*, 2000:31–46.

[7] Andrew Goldberg, Haim Kaplan, and F. Renato Werneck. Reach for A$*$: Efficient point-to-point shortest path algorithms. *IN WORKSHOP ON ALGORITHM ENGINEERING & EXPERIMENTS*, 2006:129–143.

[8] J. Halpern. Shortest route with time dependent length of edges and limited delay possibilities in nodes. *Zeitschrift für Operations Research*, 21(3):117–124, 1977.

[9] Horst W. Hamacher, Stefan Ruzika, and Stevanus A. Tjandra. Algorithms for time-dependent bicriteria shortest path problems. *Discrete Optimization*, 3(3):238–254, 2006.

[10] Tomas Jurik, Arben CELA, Rehda Hamouche, Abdellatif Reama, Rene Natowicz, Silviu-Iulian Niculescu, Christophe Villedieu, and Denis Pachetau. Energy Optimal Real-Time Navigation System: Application to a Hybrid Electrical Vehicle. *ITSC 2013*, pages 1947–1952, 2013.

[11] Emil Klafszky. Determination of shortest path in a network with time-dependent edge-lengths 1. *Mathematische Operationsforschung Statistik*, 3(4):255–257, 1972.

[12] S. Kluge, C. Santa, S. Dangl, S. Wild, M. Brokate, K. Reif, and F. Busch. On the computation of the energy-optimal route dependent on the traffic load in Ingolstadt. *Transportation Research Part C: Emerging Technologies*, 36:97–115, 2013.

[13] Sebastian Kluge. *On the computation of fuel-optimal paths in time-dependent networks*. Technical University of Munich, dissertation, 2011.

[14] Michael M. Kostreva and Malgorzata M. Wiecek. Time Dependency in Multiple Objective Dynamic Programming. *Journal of Mathematical Analysis and Applications*, 173(1):289–307, 1993.

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

[16] Ariel Orda and Raphael Rom. Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length. *Journal of the ACM*, 37(3):607–625, 1990.

[17] Ariel Orda and Raphael Rom. Minimum weight paths in time-dependent networks. *Networks*, 21(3):295–319, 1991.

[18] Stefano Pallottino and MariaGrazia Scutellà. Shortest Path Algorithms In Transportation Models: Classical and Innovative Aspects. In Patrice Marcotte and Sang Nguyen, editors, *Equilibrium and Advanced Transportation Modelling*, Centre for Research on Transportation, pages 245–281. Springer US, 1998.

[19] Peter Sanders and Dominik Schultes. Highway Hierarchies Hasten Exact Shortest Path Queries. In David Dutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, Gerhard Weikum, Gerth Stølting Brodal, and Stefano Leonardi, editors, *Algorithms – ESA 2005*, volume 3669 of *Lecture Notes in Computer Science*, pages 568–579. Springer Berlin Heidelberg, 2005.

[20] Martin Treiber, Arne Kesting, and Christian Thiemann. *How Much does Traffic Congestion Increase Fuel Consumption and Emissions? Applying a Fuel Consumption Model to the NGSIM Trajectory Data*. transportation research board, Presentation No. 08-2715, 2008.

[21] Kai Zhang, Stuart Batterman, and François Dion. Vehicle emissions in congestion: Comparison of work zone, rush hour and free-flow conditions. *Atmospheric Environment*, 45(11):1929–1939, 2011.