Modeling and Engineering Constrained Shortest Path Algorithms for Battery Electric Vehicles

Moritz Baum[1], Julian Dibbelt[2], Dorothea Wagner[3], and Tobias Zündorf[*4]

1 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
`moritz.baum@kit.edu@kit.edu`
2 **Mountain View, CA, USA**
`algo@dibbelt.de`
3 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
`dorothea.wagner@kit.edu`
4 **Karlsruhe Institute of Technology, Karlsruhe, Germany**
`tobias.zuendorf@kit.edu`

Abstract

We study the problem of computing constrained shortest paths for battery electric vehicles. Since battery capacities are limited, fastest routes are often infeasible. Instead, users are interested in fast routes where the energy consumption does not exceed the battery capacity. For that, drivers can deliberately reduce speed to save energy. Hence, route planning should provide both path *and* speed recommendations. To tackle the resulting $\mathcal{NP}$-hard optimization problem, previous work trades correctness or accuracy of the underlying model for practical running times. In this work, we present a novel framework to compute *optimal* constrained shortest paths for electric vehicles that uses more realistic physical models, while taking speed adaptation into account. Careful algorithm engineering makes the approach practical even on large, realistic road networks: We compute optimal solutions in less than a second for typical battery capacities, matching performance of previous inexact methods. For even faster performance, the approach can easily be extended with heuristics that provide high quality solutions within milliseconds.

1998 ACM Subject Classification G.2.2 Graph Theory, G.2.3 Applications

Keywords and phrases electric vehicles, constrained shortest paths, algorithm engineering

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.11

1 Introduction

Battery electric vehicles (EVs) have matured, giving the prospect of high powertrain efficiency and independence of fossil fuels, but a major hindrance of their adoption remains the limited battery capacity of most vehicles combined with a lengthy recharge time. To overcome *range anxiety*, careful route planning that prevents battery depletion during a ride is paramount. Besides a limited cruising range, another substantial difference to vehicles run by combustion engines is the ability to recuperate energy when braking. Naturally, such aspects have to be reflected in any kind of route planning application for EVs.

Classic route planning approaches make use of a graph-based representation of the considered transportation network, where scalar edge weights correspond to, e. g., travel times. A shortest path is then found by Dijkstra's algorithm [15]. A wide range of *speedup techniques* [3]

* Supported by DFG Research Grant WA 654/23-1.

enable provably correct but faster queries in practice. For instance, A* Search [27] uses *vertex potentials* to guide the search towards the target. Contraction Hierarchies (CH) [23], on the other hand, employs a preprocessing step to obtain a directed acyclic search graph that allows to skip vast parts of the network at query time. For that, it iteratively *contracts* vertices according to a heuristic vertex ranking, while adding *shortcut* edges to maintain distances within the remaining graph. Extensions to multicriteria scenarios exist for both A* [17, 32, 33, 36] and CH [21, 22, 38]. Moreover, CH and A* can be combined to Core-ALT [6], where all but the highest-ranked vertices are contracted, which form the *core* graph. On that, a variant of A* uses precomputed distances to *landmark* vertices [24].

Route planning for EVs requires handling battery capacity constraints and negative edge weights (due to recuperation), which is tractable when optimizing energy consumption as a single criterion [9, 16, 35]. However, energy-optimal routes often exhibit disproportionate detours, as using minor, slow roads can save energy due to less air drag [9]. Variants of the $\mathcal{NP}$-hard *Constrained Shortest Path (CSP)* problem [26] overcome this by minimizing energy consumption without exceeding a given time limit [37] or finding the fastest route that does not exceed battery constraints [7, 41]. Yet, time–consumption tradeoffs are not only affected by choice of route but also by driving behavior. Assuming a single, fixed speed per road segment neglects attractive solutions that may still use major roads (e. g., motorways), saving energy by deliberately driving below posted speed limits, instead. Sampling such alternative speeds, tradeoffs can be modeled by parallel edges [8, 25], but this yields too many nondominated intermediate solutions, growing exponentially even for chains of vertices. Accordingly, only heuristics offer acceptable performance for common vehicle ranges [8, 25]. By discretizing a continuous range of possible speeds, the approach has further undesirable effects: The majority of its many intermediate solutions offers insignificant tradeoffs [8], while interesting solutions are lost to the discretization; adding degree-two vertices (commonly included for visualization) affects the solution space, even when distributing speeds and consumption evenly. Instead, Hartmann and Funke [28] model tradeoffs as continuous *functions* per edge, assuming the driver can go at *any* speed within limits. Yet, for that model they propose only a heuristic extension of CH that requires minutes to answer queries on large networks. Lv et al. [31] use dynamic programming to plan the speed of a solar-powered EV, but their approach aims at simulation and is too slow for interactive applications.

Contribution and Outline. We study a generalization of the CSP problem to capture the characteristics of EVs, considering *continuous, adaptive speeds*: We allow the EV to adjust its speed to reach its target quickly and with sufficient state of charge (SoC). Using realistic consumption models, we obtain for each road segment a function mapping travel time to energy consumption, yielding a challenging, more precise problem setting (Section 2). As a first solution, we propose an exponential-time extension of Dijkstra's algorithm: By propagating continuous consumption functions during network exploration, we greatly improve performance *and* solution quality over previous discretized approaches (Section 3). We also incorporate techniques based on A* and CH, for which a particular challenge is the computation of shortcuts that represent *bivariate functions* to capture the constraints of our model (Section 4). Our experimental evaluation (Section 5) reveals that we can compute *optimal* solutions in well below a second for typical battery capacities and less than a minute for large battery capacities, on par or faster than previous *heuristic* algorithms. Our own heuristic variant provides high-quality solutions and is fast enough for interactive applications.

2 Model and Problem Statement

We use directed graphs $G = (V, E)$ to model road networks, where edges $e \in E$ represent road segments. For each, we assume that a given *tradeoff function* $g_e \colon \mathbb{R}_{>0} \to \mathbb{R}$ maps desired driving time $x \in \mathbb{R}_{>0}$ along e to energy consumption $g_e(x)$. Consumption can be negative, due to recuperation. In reality, driving time cannot be chosen arbitrarily: Lower bounds are induced by speed limits and the vehicle's maximum speed. On the other hand, driving slower than a reasonable minimum speed would mean to become an obstacle for other drivers. This yields minimum and maximum driving times $\underline{\tau} \in \mathbb{R}_{>0}$ and $\bar{\tau} \in \mathbb{R}_{>0}$, respectively, for g_e. We incorporate them into a *consumption function* $c_e \colon \mathbb{R}_{\geq 0} \to \mathbb{R} \cup \{\infty\}$ with

$$c_e(x) := \begin{cases} \infty & \text{if } x < \underline{\tau}, \\ g_e(\bar{\tau}) & \text{if } x > \bar{\tau}, \\ g_e(x) & \text{otherwise.} \end{cases}$$

Thus, driving times below $\underline{\tau}$ are infeasible (modeled as infinite consumption) and driving times above $\bar{\tau}$ become unprofitable. In the special (degenerate) case of $\underline{\tau} = \bar{\tau}$, the function c_e represents a constant pair $(\underline{\tau}, c_e(\underline{\tau}))$ of driving time and consumption. We then call c_e *constant*, as the edge e allows no speed adaptation.

Further, the EV is equipped with a battery that has a *capacity* $M \in \mathbb{R}_{\geq 0}$. The SoC must not drop below 0 nor exceed M. Incorporating these constraints, we obtain a bivariate *SoC function* $f_e \colon \mathbb{R}_{\geq 0} \times [0, M] \cup \{-\infty\} \to [0, M] \cup \{-\infty\}$ for every $e = (u, v) \in E$, mapping SoC at u to SoC at v when traversing e with a specific driving time. It is given by

$$f_e(x, b) := \begin{cases} -\infty & \text{if } b - c_e(x) < 0, \\ M & \text{if } b - c_e(x) > M, \\ b - c_e(x) & \text{otherwise,} \end{cases}$$

where an SoC of $-\infty$ denotes an empty battery. Hence, $f_e(x, b) = -\infty$ means that the edge cannot be traversed at the corresponding speed (as the battery would run empty).

An *s–t path* in G is a sequence $P = [s = v_1, v_2 \ldots, v_k = t]$ of vertices with $(v_i, v_{i+1}) \in E$ for $1 \leq i \leq k - 1$. If $s = t$, we call P a *cycle*. Given the SoC $b_s \in [0, M]$ at s, we obtain a corresponding SoC at t by iteratively picking driving times $x_i \in \mathbb{R}_{\geq 0}$ (starting at s) and evaluating the SoC function $f_{(v_i, v_{i+1})}$ for x_i and the SoC at v_i. Due to physical constraints, we presume that for cycles this procedure never increases the SoC at $s = t$. For paths $P = [v_1, \ldots v_i]$ and $Q = [v_i, \ldots, v_k]$, $P \circ Q := [v_1, \ldots, v_i, \ldots, v_k]$ is their concatenation.

Given a source $s \in V$, a target $t \in V$, and an initial SoC $b_s \in [0, M]$, the *Electric Vehicle Constrained Shortest Path (EVCSP)* Problem is to find an s–t path $P = [s = v_1, v_2 \ldots, v_k = t]$ together with driving times $x_i, i \in \{1, \ldots, k - 1\}$, for every edge in P that respect battery constraints and minimize overall travel time $x := \sum_{i=1}^{k-1} x_i$ in G. This yields an $\mathcal{NP}$-hard problem by reduction from CSP [26]. An instance of CSP corresponds to an instance of EVCSP where all functions are degenerate constant tuples with nonnegative consumption.

A Simplified Model. We illustrate SoC functions in an example using simplistic but vivid tradeoff functions. For now, let tradeoff functions be *decreasing* and *linear*, i. e., $g_e(x) = \alpha x + \beta$ for every $e \in E$, where $\alpha \in \mathbb{R}_{\leq 0}$ and $\beta \in \mathbb{R}$ are constant coefficients. The values α and β may differ between edges to reflect different road types or other relevant factors [12, 42]. Figures 1a and 1b show consumption functions (plugging in limits $\underline{\tau}$ and $\bar{\tau}$ on driving time) for two edges (u, v) and (v, w). We are interested in the consumption function of the path

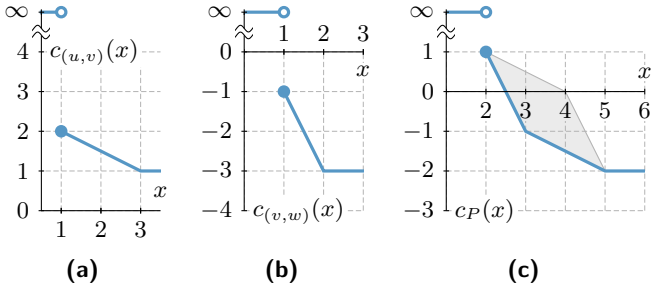

Figure 1 Consumption functions based on a simple model. (a) Function $c_{(u,v)}$ of an edge (u, v) with $\underline{\tau} = 1$ and $\bar{\tau} = 3$. (b) Function $c_{(v,w)}$ of an edge (v, w) with $\underline{\tau} = 1$ and $\bar{\tau} = 2$. (c) The function c_P of $P = [u, v, w]$. The shaded area indicates possible pairs of driving time and consumption.

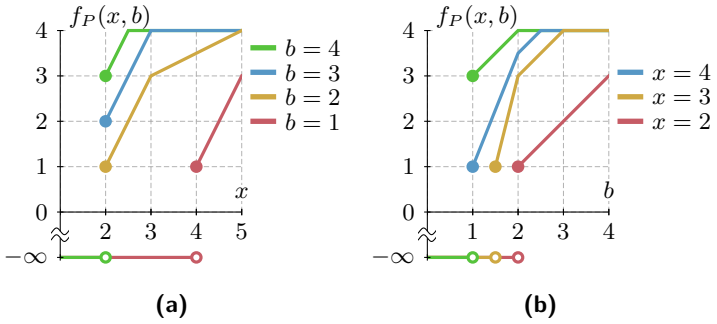

Figure 2 The bivariate SoC function of the path P from Figure 1, for $M = 4$. (a) The SoC f_P at v, subject to driving time x on P for different fixed values b of initial SoC. (b) The SoC f_P at v, subject to initial SoC b for different fixed values x of driving time.

$P = [u, v, w]$, i. e., a function c_P that maps driving time x spent on P to *minimum* energy consumption $c_P(x)$. Formally, to get $c_P(x)$ for a driving time $x \in \mathbb{R}_{\geq 0}$, we must pick values $x_1 \in \mathbb{R}_{\geq 0}$ and $x_2 \in \mathbb{R}_{\geq 0}$, such that $x = x_1 + x_2$ and $c_{(u,v)}(x_1) + c_{(v,w)}(x_2)$ is minimized. Figure 1c shows possible distributions of driving times among the two edges and the resulting energy consumption. Their lower envelope yields the desired function c_P. Intuitively, we want to spend as much of the available time as possible on the edge that provides the better tradeoff for saving the most energy, i. e., the function with steeper slope. As a result, the consumption function of a path is always *convex* on its finite imaginary part. Moreover, while tradeoff functions of edges are linear in the interval $[\underline{\tau}, \bar{\tau}]$ of admissible driving times, the tradeoff function of a path is *piecewise* linear within its corresponding interval.

When considering battery constraints, energy consumption depends not only on driving time but also on initial SoC. Note that consumption is positive on (u, v) and negative on (v, w). As before, the edge (v, w) provides the better tradeoff. However, for low initial SoC, we must ensure that (u, v) can be traversed first, spending additional time on this edge in order to obtain a feasible solution at all. In contrast, high initial SoC values may prevent recuperation along (v, w), limiting the payoff of driving slower. Figure 2 illustrates the resulting *bivariate* SoC function f_P for specific values of initial SoC and driving time.

A Realistic Model. In this work, we use a more realistic model, detailed below. Both driving time and energy consumption depend on the vehicle's speed. In accordance with realistic physical models [1, 2, 10, 18, 28, 30, 31], we assume that energy consumption on a

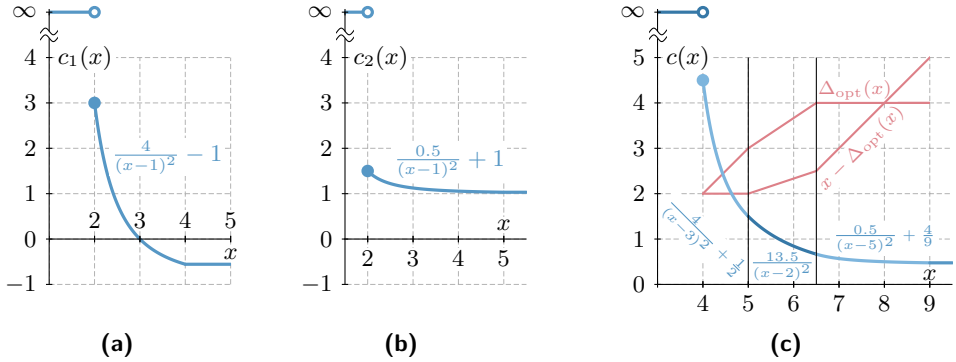

Figure 3 Linking consumption functions. (a) Function c_1 with $\alpha_1 = 4$, $\beta_1 = 1$, $\gamma_1 = -1$, $\underline{\tau} = 2$, and $\bar{\tau}_1 = 4$. (b) Function c_2 with $\alpha_1 = 0.5$, $\beta_1 = 1$, $\gamma_1 = 1$, $\underline{\tau}_1 = 2$, and $\bar{\tau}_1 = 5$. (c) Function $c = \text{link}(c_1 c_2)$, with $c(x) = c_1(\Delta_{\text{opt}}(x)) + c_2(x - \Delta_{\text{opt}}(x))$. It is defined by three subfunctions with subdomains $[4, 5]$, $[5, 6.5]$, $[6.5, 9]$. Values $\Delta_{\text{opt}}(x)$ and $x - \Delta_{\text{opt}}(x)$ indicate the share of c_1 and c_2.

road segment $e \in E$ is expressed by a function $h_e \colon \mathbb{R}_{>0} \to \mathbb{R}$ with $h_e(v) = \lambda_1 v^2 + \lambda_2 s_e + \lambda_3$, where $v \in \mathbb{R}_{>0}$ is the (constant) vehicle speed, $s_e \in \mathbb{R}$ is the (constant) slope of the road segment, and $\lambda_1 \in \mathbb{R}_{\geq 0}$, $\lambda_2 \in \mathbb{R}_{\geq 0}$, and $\lambda_3 \in \mathbb{R}_{\geq 0}$ are constant nonnegative coefficients of the consumption model (all values may vary for different edges). Note that assuming constant speed and slope per edge is not a restriction, as intermediate vertices can be added to model changing conditions. Further, one can show that varying the speed on a single road segment (with constant slope and speed limit) never pays off in our model [28, Corollary 1].

As we are interested in functions mapping *driving time* $x \in \mathbb{R}_{>0}$ to energy consumption $g_e(x)$, we substitute $v = \ell_e / x$, where ℓ_e is the length of the road segment. Slope and length of an edge are fixed, so we simplify this by setting $\alpha := \lambda_1 / \ell_e^2$ and $\gamma := \lambda_2 s_e + \lambda_3$. Observe that $\alpha \in \mathbb{R}_{\geq 0}$ is nonnegative, while $\gamma \in \mathbb{R}$ may be negative (for downhill edges). We introduce a third constant $\beta \in \mathbb{R}_{\geq 0}$, needed later to shift functions along the time axis. Altogether, we obtain the tradeoff function $g_e \colon \mathbb{R}_{>0} \to \mathbb{R}$ with

$$g_e(x) := \frac{\alpha}{(x - \beta)^2} + \gamma. \tag{1}$$

For single edges, we always obtain $\beta = 0$ and assume driving time x to be *strictly* positive. Thus, the denominator $x - \beta$ is strictly positive and $g_e(x)$ is finite. Further, g_e is *decreasing* and *convex* on $\mathbb{R}_{>0}$ in this case. In the simplistic model discussed above, we have seen that tradeoff functions of *paths* may be piecewise linear. Similarly, we allow tradeoff functions in the realistic model to be defined *piecewise*, so they may consist of multiple subfunctions of the form in Equation 1. Tradeoff functions of paths may also use values $0 < \beta < x$ to reflect additional time spent on previous edges. Plugging in the values $\underline{\tau} \in \mathbb{R}_{>0}$ and $\bar{\tau} \in \mathbb{R}_{>0}$, we obtain the *consumption function* $c_e \colon \mathbb{R}_{\geq 0} \to \mathbb{R} \cup \{\infty\}$.

3 Basic Approach

We generalize the (exponential-time) bicriteria variant [34] of Dijkstra's algorithm [15] to solve EVCSP. As a crucial ingredient, the algorithm requires a *link operation*: For two consumption functions c_1 and c_2 modeling consumption on two paths P_1 and P_2, the function $c := \text{link}(c_1, c_2)$ maps driving time spent on $P := P_1 \circ P_2$ to *minimum* possible energy consumption (bar battery constraints). Let $\underline{\tau}_1$, $\bar{\tau}_1$, $\underline{\tau}_2$, and $\bar{\tau}_2$ denote the respective minimum

and maximum driving times of c_1 and c_2. We obtain $c(x) = \infty$ for all $x < \underline{\tau}_1 + \underline{\tau}_2$ and $c(x) = c_1(\bar{\tau}_1) + c_2(\bar{\tau}_2)$ for $x > \bar{\tau}_1 + \bar{\tau}_2$. For all $x \in [\underline{\tau}_1 + \underline{\tau}_2, \bar{\tau}_1 + \bar{\tau}_2]$, we have to compute

$$c(x) = \min_{\substack{\Delta \in [\underline{\tau}_1, \bar{\tau}_1] \\ \Delta \in [x - \bar{\tau}_2, x - \underline{\tau}_2]}} c_1(\Delta) + c_2(x - \Delta).$$

In other words, we have to divide the amount of time that exceeds the minimum possible total driving time among the two paths such that consumption is minimized; see Figure 3 for an example. Although realistic functions require a more technical analysis, many observations made for our simplistic (linear) model from the previous section carry over to the more realistic (nonlinear) tradeoff functions. In fact, the function c can be computed in linear time in the number of subfunctions defining c_1 and c_2.

Algorithm Description. Given a source $s \in V$, a target $t \in V$, and initial SoC $b_s \in [0, M]$, the *tradeoff function propagating (TFP)* algorithm solves EVCSP. It propagates labels consisting of consumption functions (defined piecewise, by sequences of tradeoff functions) and applies battery constraints on-the-fly. Hence, it does not have to maintain bivariate SoC functions explicitly. The algorithm starts with the constant label $c_s \equiv M - b_s$ at s. The label is also added to a priority queue, which uses minimum driving time of a label as key. In each step of its main loop, the algorithm extracts and *settles* a label c_u (at some vertex $u \in V$) with minimum key from the queue. For every edge $(u, v) \in E$, the function $c := \text{link}(c_u, c_{(u,v)})$ is computed. Note that c may violate battery constraints, so we set $c(x) := \infty$ for all $x \in \mathbb{R}_{\geq 0}$ with $c(x) > M$ and $c(x) := 0$ for all $x \in \mathbb{R}_{\geq 0}$ with $c(x) < 0$. The resulting function is added to the priority queue, unless it is *dominated* by existing labels at v; we say that a label c_1 dominates another label c_2 if $c_1(x) \leq c_2(x)$ for all $x \in \mathbb{R}_{\geq 0}$.

To keep the number of label comparisons low, each vertex $v \in V$ maintains a set $L_{\text{set}}(v)$ and a heap $L_{\text{uns}}(v)$ containing its *settled* and *unsettled* labels, respectively. We maintain the invariant that for each $v \in V$, the unsettled label in $L_{\text{uns}}(v)$ with *minimum* key is not dominated by any settled label in $L_{\text{set}}(v)$. Labels (at v) added to the priority queue are also pushed into $L_{\text{uns}}(v)$. Every time the minimum element of $L_{\text{uns}}(v)$ changes (because an element is added or extracted), we check whether the new minimum element is dominated by any settled label in $L_{\text{set}}(v)$ and discard it in this case [7]. Dominance is tested as follows. For two *subfunctions* (with the form of Equation 1), we can test in constant time whether one dominates the other (by evaluating extreme points of their difference and subdomain borders). For piecewise-defined consumption functions, we exploit that we only need to compare subfunctions whose subdomains intersect. This allows us to test for dominance in a linear scan (comparing subfunctions in increasing order of driving time). Given a consumption function c in the set $L_{\text{uns}}(v)$ of some vertex $v \in V$, a naïve implementation then performs pairwise comparisons to functions in $L_{\text{set}}(v)$ to determine whether c is dominated by any of them. In doing so, the algorithm may miss cases where c is merely *partially* dominated, or dominated only by the lower envelope of *several* functions. Although including dominated labels in $L_{\text{set}}(v)$ does not affect correctness, it may lead to unnecessary vertex scans and increases the label size. Instead of pairwise dominance checks, we therefore identify dominated parts of c in a single coordinated scan over c and *all* functions in $L_{\text{set}}(v)$.

TFP is *label setting*, i. e., labels extracted from the queue are never dominated later on. An optimal (constrained) path is found once a label at t is extracted, which gives the optimal driving time. It is also possible to retrieve the optimal path and driving speeds.

A Polynomial-Time Heuristic. To improve running times, TFP can easily be extended to a heuristic search, at the cost of inexact results. We propose a polynomial-time approach based on ε-dominance [4]. When testing dominance of a label $c \in L_{\text{uns}}(v)$ at some vertex $v \in V$,

it is kept in $L_{\text{uns}}(v)$ only if it yields an improvement (over labels in $L_{\text{set}}(v)$) by at least a certain fraction εM, with $\varepsilon \in (0, 1]$, for some driving time. Hence, we test for every $x \in \mathbb{R}_{\geq 0}$ whether $c(x) + \varepsilon M \leq c_{\text{set}}(x)$ holds for *all* settled functions $c_{\text{set}} \in L_{\text{set}}(v)$. Then, the number of settled labels per set can become at most $\lceil 1/\varepsilon \rceil$, which yields polynomial running time.

4 Speedup Techniques

We propose speedup techniques based on A* and CH for TFP (and its heuristic variant). Combining both techniques, we obtain our fastest variant, *CHAsp (CH, A*, Adaptive Speeds)*. Our techniques do not alter the output of the algorithm, so correctness of TFP is maintained.

A* Search. This well-known technique [27, 33] uses a *potential function* $\pi\colon V \to \mathbb{R}_{\geq 0}$. The potential $\pi(v)$ of a vertex $v \in V$ is added to all keys of labels when running TFP, so labels are extracted in a different order. We compute the potential function at query time.

Our first variant uses a cost functions $\underline{d}\colon E \to \mathbb{R}_{\geq 0}$ with $\underline{d}(e) = c_e(\underline{\tau}_e)$, i. e., minimum driving time on an edge. Before running TFP, a *backward* search (i. e., Dijkstra's algorithm traversing edges in backward direction) from the target t computes, for each vertex $v \in V$, the minimum *unconstrained* driving time $\underline{d}(v, t)$ from v to the t. We obtain a consistent potential function $\pi_d\colon V \to \mathbb{R}_{\geq 0}$ by setting $\pi_d(v) := \underline{d}(v, t)$ [40]. Similarly, we compute lower bounds on energy consumption, which allow us to prune the TFP search [8].

The potential function $\pi_d(v)$ may be too conservative if consumption on the optimal path is very high. In such cases, it pays off to use a potential function $\pi_f\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$ that incorporates current SoC at a vertex [7]. We represent $\pi_f(v, b)$ with a convex, piecewise linear function that maps SoC $b \in [0, M]$ at a vertex $v \in V$ to a lower bound on remaining driving time. The functions are determined in a label-correcting backward search from t.

Contraction Hierarchies. We propose an adaptation of CH to our scenario, which adds a preprocessing step for faster queries. As in plain CH [23], vertices are contracted iteratively (ordered by heuristic *rank*) during preprocessing and *shortcut edges* are added to maintain distances. However, we contract only a subset of the vertices, leaving an uncontracted *core* graph – a common approach in complex scenarios [7, 14, 28, 37]. Since the SoC at a vertex $u \in V$ is only known at query time in our setting, any shortcut (u, v) has to store a bivariate SoC function $f_{(u,v)}$. Figure 4 illustrates how the initial SoC influences energy consumption in our model. Their bivariate nature makes explicit construction and comparison of SoC functions rather challenging. We discuss simple representations of SoC functions in certain cases, exploiting that most consumption values are positive in realistic instances. We say that a path P is *discharging* if the SoC on P never exceeds the (arbitrary) initial SoC, i. e., there is no prefix of P that has negative minimum consumption for arbitrary driving times (subpaths with negative consumption are allowed, though). Hence, it is not necessary to explicitly check whether the SoC exceeds M on a discharging path. We show how the SoC function of a discharging path is represented by at most two consumption functions.

As a first example, assume we are given a path $P = P_1 \circ P_2$ consisting of two subpaths P_1 and P_2 with respective consumption functions c_1 and c_2, as in Figure 4. Let $\underline{\tau}_1, \bar{\tau}_1, \underline{\tau}_2, \bar{\tau}_2$ denote their corresponding minimum and maximum driving times. Assume that $c_1(x) > 0$ is *positive* for all $x \in \mathbb{R}_{\geq 0}$, while $c_2(x) \leq 0$ is *nonpositive* for all $x \in [\underline{\tau}_2, \infty)$. Finally, assume that $|c_1(\bar{\tau}_1)| \geq |c_2(\bar{\tau}_2)|$, i. e., the cost of P_1 is higher than the gain of P_2 for *any* driving time, so P is discharging. To derive the SoC function of P we introduce two auxiliary functions: a *positive part* c^+ with $c^+(x) := c_1(x - \underline{\tau}_2)$, and a *negative part* c^- with $c^-(x) := c_2(x + \underline{\tau}_2)$.

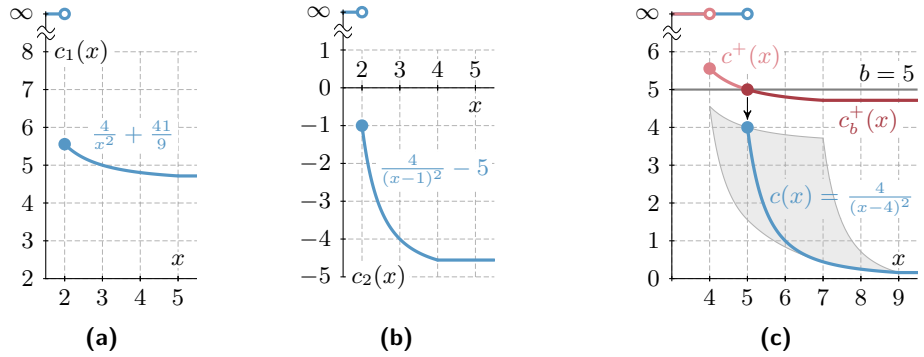

Figure 4 Constructing a consumption function depending on initial SoC. (a) Function c_1 of a path P_1. (b) Function c_2 of a path P_2. (c) Due to battery constraints, the minimum driving time on $P = P_1 \circ P_2$ is 5 for an initial SoC $b = 5$. This yields the consumption function $c = \text{link}(c_b^+, c^-)$. The shaded area indicates possible values of consumption functions for different values of initial SoC.

The original functions are shifted along the x-axis to simplify the analysis (note that the minimum feasible driving time of c^- is 0). Given some initial SoC $b \in [0, M]$, the positive part c^+, and the negative part c^-, we first define the *constrained positive part* c_b^+ as

$$c_b^+(x) := \begin{cases} \infty & \text{if } b < c^+(x) \\ c^+(x) & \text{otherwise,} \end{cases}$$

which applies battery constraints along P_1 for an initial SoC of b; see Figure 4. Then, the SoC function f_P of the path P evaluates to $f_P(x, b) = b - \text{link}(c_b^+, c^-)(x)$ for arbitrary $x \in \mathbb{R}_{\geq 0}$ and $b \in [0, M]$. The function first applies battery constraints on the positive part and links the resulting function with the negative part.

We now describe how SoC functions representing general discharging paths are constructed from two given SoC functions of discharging paths. Assume we are given a discharging path P_1 whose SoC function is defined by two consumption functions c_1^+ and c_1^-, as described above. Similarly, we are given a discharging path P_2 with respective consumption functions c_2^+ and c_2^-. Observe that the path $P := P_1 \circ P_2$ must be discharging as well. Apparently, if we know the initial SoC, we can compute energy consumption on P by computing $\text{link}(\text{link}(\text{link}(c_1^+, c_1^-)c_2^+)c_2^-)$ and applying battery constraints *before* each link operation, like in the TFP algorithm. However, we want to represent P with only two consumption functions c^+ and c^-. Recall that the only constraint we have to check for discharging paths is whether the SoC drops below 0. Thus, we identify a new positive part c^+ as follows. Since both c_1^- and c_2^- are nonpositive for all admissible driving times, the constraint needs only to be checked for c_1^+ and c_2^+ (i. e., before the first and third link operation). To integrate these checks into a single positive part c^+, we first compute the function $h := \text{link}(c_1^-, c_2^+)$. Clearly, the battery can only run empty on P_2 if this consumption function is positive for some admissible driving time. To distinguish this case, we split h into a positive part h^+ with $h^+(x) := \max\{h(x), 0\}$ and a negative part h^- with $h^-(x) := h(x)$ if $h(x) \leq 0$ and $h^-(x) := \infty$ otherwise. Since h is a decreasing consumption function, so are h^+ and h^-. We obtain the positive part c^+ of P by setting $c^+(x) := \text{link}(c_1^+, h^+)(x - \underline{\tau})$ and the negative part c^- by setting $c^-(x) := \text{link}(h^-, c_2^-)(x + \underline{\tau})$, where $\underline{\tau}$ is the minimum driving time of h^-. The SoC function of P is obtained from c^+ and c^- as described above.

During preprocessing, we only allow a vertex $v \in V$ to be contracted if all new shortcuts created as part of its contraction are discharging. We call v *active* in this case. Note that the number of active vertices grows as contraction proceeds, as contraction produces longer

Table 1 Benefits of our approach (Eur-PG, 2 kWh). For TFP and TFP-dom. (improved dominance tests), we report the number of settled labels (# Lbls.), number of label comparisons during the forward search (# Dom.), average and maximum running times, and relative driving time savings over the constrained path found by BSP on discretized speeds.

	Query				Path Savings	
Algo.	# Lbls.	# Dom.	avg. [ms]	max. [ms]	avg. [%]	max. [%]
BSP	30 990 276	21 300 657 522	47 755	779 756	–	–
TFP	103 119	4 399 002	444	14 347	2.7 %	9.4 %
TFP-dom.	46 228	700 546	103	3 851	2.7 %	9.4 %

shortcuts, which are more likely to consist of long positive parts. Since we deal with a bicriteria scenario, vertex contraction may produce multi-edges. In such cases, we only want to keep shortcuts whose SoC functions are not dominated by parallel shortcuts. Hence, after contraction of a vertex, we delete (parts of) SoC functions of shortcut candidates that are dominated by existing functions between the same pair of vertices (and vice versa). To this end, we derive efficient dominance checks for (simple) bivariate SoC functions that can be performed in linear time (in the number of subfunctions of all involved functions). Finally, before adding a (nondominated) shortcut candidate to the graph, we run a witness search [23] to test if the shortcut is necessary to maintain distances. As an exact approach would require propagation and comparison of bivariate SoC functions, our witness search computes univariate *upper bounds* on energy consumption instead. This does not violate correctness, but may result in unnecessary shortcuts.

Queries. Plain CH uses a bidirectional search, which scans only edges to vertices of higher rank in the input graph enriched with shortcuts obtained during preprocessing. In our case, however, the SoC at the target vertex $t \in V$ is not known at query time, which makes backward search difficult. Instead, we extract the search space in a (backward) BFS from t, scanning and marking only edges to vertices of higher rank. Afterwards, we execute TFP from the source vertex s, scanning upward edges (with respect to ranks of incident vertices), core edges, and marked downward edges. For faster queries, we can combine this search with A*.

5 Experiments

We implemented all approaches in C++, using g++ 4.8.3 (-O3) as compiler. Experiments were conducted on a single core of a 4-core Intel Xeon E5-1630v3 clocked at 3.7 GHz, with 128 GiB of DDR4-2133 RAM, 10 MiB of L3 cache, and 256 KiB of L2 cache.

We consider road networks of Europe with 22 198 628 vertices and 51 088 095 edges and Germany with 4 692 091 vertices and 10 805 429 edges, provided by PTV AG (`http://ptvgroup.com`). Combining reasonable minimum speeds for different road types (e. g., 80 km/h on motorways and 30 km/h in residential areas) with the posted speed limits (if higher), we get intervals of allowed speeds per road segment, resulting in 25 % and 38 % of nonconstant edges for Germany and Europe, respectively. Applying elevation data from the Shuttle Radar Topography Mission, v4.1 (`srtm.csi.cgiar.org`), we derived realistic energy consumption from two detailed micro-scale emission models [29]: one based on a Peugeot iOn and one artificial model [39] that additionally accounts for auxiliary consumers (e. g., air conditioning). These data sources are proprietary, but enable evaluation on

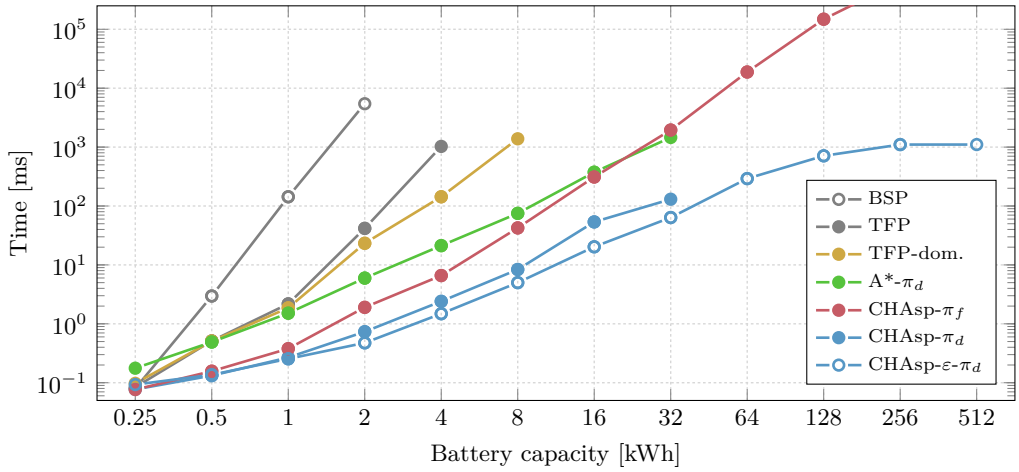

Figure 5 Scalability of BSP, our TFP algorithm, TFP with improved dominance tests (TFP-dom.), speedup techniques (A*-π_d, CHAsp-π_d, and CHAsp-π_f), and our heuristic approach CHAsp-ε-π_d with $\varepsilon := 0.1$. A capacity of 512 kWh corresponds to a range of roughly 3 000 km.

detailed and realistic input data. We denote our instances by Germany-Aux (Ger-AX), Germany-Peugeot (Ger-PG), Europe-Aux (Eur-AX), and Europe-Peugeot (Eur-PG). They have negative consumption (for at least some driving times) on 7.8 % (Ger-AX) to 12.9 % (Eur-PG) of their edges.

For comparison, we consider parallel edges and bicriteria shortest paths (BSP) [34] to model adaptive speeds, as was best practice in previous approaches [8]. We generate multi-edges by sampling consumption functions at discrete velocity steps of 10 km/h.

We evaluate random *in-range* queries, i. e., we pick a source vertex $s \in V$ uniformly at random. Among all vertices in range from s with an initial SoC $b_s = M$, we pick the target vertex $t \in V$ uniformly at random. Since unreachable targets are easily detected by backward search phases of A* (or any algorithm for computing energy-optimal routes [9, 16, 35]), this yields more challenging and interesting queries for us.

Model Validation and Scalability. We have argued that an approach based fully on consumption *functions* unlocks both better tractability and improved solution quality compared to discrete speeds and BSP. Indeed, we observe a significant speedup by simply switching to our more realistic model, as Table 1 shows. TFP is up to two orders of magnitudes faster than BSP and finds paths that are up to 9.4 % quicker (within SoC constraints), since it evaluates speed–consumption tradeoffs more fine-granularly while maintaining less query state (labels of continuous functions expressed by few parameters instead of large, discrete Pareto sets). This is interesting, as sampling was expressly considered to manage tractability [8, 25, 28]. In fact, even though atomic operations (linking and comparing labels) are more expensive for TFP, a drastic reduction in the number of vertex scans explains the speedup.

Figure 5 gives an overview of our approaches and their scalability across increasing battery capacities. For each capacity, we ran 100 random in-range queries, reporting median running time if all 100 queries terminated within one hour. Beyond the previously discussed BSP, TFP, and TFP-dom., A* enables reasonable running times for capacities of up to 32 kWh, without any preprocessing. Adding preprocessing, CHAsp-π_d provides further speedup by about an order of magnitude. In comparison, median running times of CHAsp-π_f are slower for all ranges up to 32 kWh. However, this algorithm is more robust against outliers and

Table 2 Impact of core size on performance (Ger-PG, 16 kWh). Vertex contraction stopped once the average degree of active vertices in the core reached a given threshold (Ø Deg). We report the resulting core size (# Vertices), preprocessing time, and average query times for 1 000 queries using CHAsp with potential functions π_d and π_f, respectively.

Core size		Prepr.	Query [ms]	
Ø Deg.	# Vertices	[h:m:s]	π_d	π_f
0	–	–	3 326.0	4 861.5
8	720 514 (15.36 %)	5:07	737.2	798.3
16	400 174 (8.53 %)	13:25	496.2	485.0
32	305 301 (6.51 %)	31:44	451.8	434.0
64	268 436 (5.72 %)	1:11:13	505.5	473.1
128	251 410 (5.36 %)	2:37:23	649.1	586.1

Table 3 Preprocessing and exact query performance for the potential functions π_d and π_f. For the ranges 16 kWh and 85 kWh, we show number of labels settled during the forward search (# Lbls.), number of label comparisons during the forward search (# Dom.) and total query times.

Prepro.			16 kWh			85 kWh		
Inst.	[h:m:s]	Algo.	# Lbls.	# Dom.	Query [ms]	# Lbls.	# Dom.	Query [ms]
Ger-AX	30:34	CHAsp-π_d	152	3 788	4.2	24 715	4 312 923	552.3
Ger-AX	30:34	CHAsp-π_f	61	448	17.0	406	11 813	1 236.7
Ger-PG	31:44	CHAsp-π_d	32 773	6 352 488	451.8	2 272 350	2 130 447 427	131 562.0
Ger-PG	31:44	CHAsp-π_f	6 008	491 173	434.0	32 182	6 836 380	14 873.5
Eur-AX	3:10:43	CHAsp-π_d	124	2 175	4.0	27 358	12 159 343	960.9
Eur-AX	3:10:43	CHAsp-π_f	73	1 006	15.8	871	46 529	1 174.7
Eur-PG	3:13:01	CHAsp-π_d	23 304	5 024 403	346.1	–	–	–
Eur-PG	3:13:01	CHAsp-π_f	6 629	800 430	341.7	105 792	44 986 403	34 617.4

is the only exact method that terminates within an hour for all queries at 64 kWh and up. Finally, our heuristic variant scales very well with vehicle range: Query times actually bottom out for large battery capacities, as the vehicle range gets close to the graph diameter.

Detailed Experiments. We evaluate different variants of our fastest approach, CHAsp. Table 2 shows CH preprocessing effort and query performance subject to core size on Ger-PG, for a common battery capacity of 16 kWh (corresponding to a range of 100 km). Contraction becomes much slower beyond a core degree of 32, which is explained by the small number of remaining active (i. e., contractable) vertices in the core. This also explains why speedup compared to the baseline (Ø deg = 0 is equivalent to plain TFP combined with A*) is much smaller than in simpler applications, where CH typically improves the baseline by several orders of magnitude [23]. Similar observations were made in other complex settings, including time-dependent [5, 11] and multicriteria [21, 22] scenarios. Nevertheless, CH still yields an improvement by up to an order of magnitude in our case. In our subsequent experiments, we pick an average core degree of 32 as stopping criterion of CH preprocessing.

Table 3 reports performance of CHAsp on all instances for capacities of 16 kWh and 85 kWh (as in Tesla models, with a range of 400–500 km). Figures are average values for 1 000 in-range queries. For 16 kWh, our techniques find the optimal solution in well below a second on average. For Ger-AX and Eur-AX, we even achieve query times in the order of milliseconds.

Table 4 Performance of the heuristic variant of CHAsp-π_d, for different choices of the parameter ε (see Section 3) on the hard instances Ger-PG and Eur-PG. We show figures on query performance for 1 000 random queries with a range of 16 kWh, as in Table 3. Additionally, we report the percentage of feasible and optimal results, as well as the average and maximum relative error.

			Query			Result Quality			
Inst.	Prepro.	ε	# Lbls.	# Dom.	T. [ms]	Feas.	Opt.	Avg.	Max.
Ger-PG	31:43	0.00	32 773	6 352 488	451.8	100.0 %	100.0 %	1.0000	1.0000
	30:41	0.01	19 922	1 949 458	225.6	100.0 %	89.4 %	1.0001	1.0047
	25:49	0.10	6 891	208 058	75.6	98.9 %	62.8 %	1.0013	1.0502
	17:48	1.00	1 742	11 149	30.7	95.1 %	47.6 %	1.0144	1.2294
Eur-PG	3:09:22	0.00	23 304	5 024 403	346.1	100.0 %	100.0 %	1.0000	1.0000
	3:04:48	0.01	12 803	1 132 685	151.6	100.0 %	82.8 %	1.0001	1.0145
	2:47:09	0.10	5 045	126 662	60.9	99.5 %	57.5 %	1.0020	1.0418
	2:14:03	1.00	1 428	7 641	28.2	92.7 %	45.8 %	1.0203	1.3960

This gap in running time is explained by the difference in the number of edges with negative cost, caused by the underlying consumption model. One could even argue that the instances Ger-PG and Eur-PG are actually rather excessive in this regard, by not accounting for any auxiliary consumers at all. As a result, these instances are significantly more difficult to solve. Regarding the potential functions π_d and π_f, the search space is consistently smaller when using π_f, but the backward search is more expensive. In fact, it becomes the major bottleneck for a battery capacity of 16 kWh on the easier instances. Consequently, query times are slower by about a factor of 4. For harder scenarios, however, the potential function π_f provides better results due to better scalability. Note that when using π_d, at least one query exceeded our threshold of one hour in computation time on Eur-PG. In summary, we can solve EVCSP *optimally* for typical ranges in less than a second, even on hard instances. For very long ranges, our algorithm computes the optimum in well below a minute on average (using π_f), despite its exponential running time.

In Table 4, we evaluate our heuristic approach for different choices of ε (in % of total SoC). During preprocessing, new shortcuts are included only if they *significantly* improve on the existing ones. Thus, preprocessing becomes faster and core sizes (not reported in the table) decrease by up to 30 %. Regarding queries, we achieve a speedup by an order of magnitude. However, the choice of ε clearly matters. For $\varepsilon = 0.01$, the decrease in quality is negligible, but speedup (about a factor of 2) is rather limited as well. For $\varepsilon = 0.1$, on the other hand, the optimal solution is still found very often. The average error is roughly 0.2 %, while the overall maximum error is 5 %, which is acceptable in practice. Finally, for $\varepsilon = 1.0$, both the average and maximum error increase significantly. Given that speedup is also limited compared to $\varepsilon = 0.1$, we conclude that the latter provides the best tradeoff in terms of quality and query performance: providing high-quality solutions, it enables query times of well below 100 ms, which is fast enough even for interactive applications. Moreover, note that in cases where no path is found (about 1 % of all queries for $\varepsilon = 0.1$), a simple fallback solution could return the energy-optimal path, which can be computed quickly [9, 16, 35].

6 Conclusion

We introduced a novel framework for computing constrained shortest paths for EVs in practice, using realistic consumption models. Our base algorithm TFP respects battery

constraints and accounts for adaptive speeds in a mathematically sounder way that unlocks *both* better query performance *and* improved solution quality when compared to previous approaches using discretized, sampled speeds. Nontrivial speedup techniques based on A* and CH make the algorithm practical. For typical EV ranges, it computes *optimal* solutions in less than a second, making it the first practical *exact* approach – with running times similar to previous inexact methods [8, 25, 28]. Our own heuristic enables even faster queries while retaining high-quality solutions.

The result of our computations is not only the suggested route from source to target but also optimal driving speeds along that route. In practice, these can be passed to the driver as recommendations or directly to a cruise control unit. With the advent of autonomous vehicles, the output of our algorithms can also be used for speed planning of self-driving EVs, either directly or after further refinement [19]. For future work, a next step would be the integration of planned charging stops [7, 37]. From a practical point of view, it might also be interesting to consider adaptive speeds only on the fastest roads (e. g., motorways), where going below the speed limit really pays off the most. Then, contracting vertices incident to *constant* edges in CH might be a promising approach. Finally, we are interested in the integration of variable speed limits imposed by, e. g., historic knowledge of traffic patterns [5, 13, 20].

References

1 Shubham Agrawal, Hong Zheng, Srinivas Peeta, and Amit Kumar. Routing Aspects of Electric Vehicle Drivers and their Effects on Network Performance. *Transportation Research Part D: Transport and Environment*, 46:246–266, 2016.

2 Johannes Asamer, Anita Graser, Bernhard Heilmann, and Mario Ruthmair. Sensitivity Analysis for Energy Demand Estimation of Electric Vehicles. *Transportation Research Part D: Transport and Environment*, 46:182–199, 2016.

3 Hannah Bast, Daniel Delling, Andrew V. Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F. Werneck. Route Planning in Transportation Networks. In *Algorithm Engineering: Selected Results and Surveys*, volume 9220 of *Lecture Notes in Computer Science*, pages 19–80. Springer, 2016.

4 Lucas S. Batista, Felipe Campelo, Frederico G. Guimarães, and Jaime A. Ramírez. A Comparison of Dominance Criteria in Many-Objective Optimization Problems. In *Proceedings of the 13th IEEE Congress on Evolutionary Computation (CEC'11)*, pages 2359–2366. IEEE, 2011.

5 Gernot V. Batz, Robert Geisberger, Peter Sanders, and Christian Vetter. Minimum Time-Dependent Travel Times with Contraction Hierarchies. *ACM Journal of Experimental Algorithmics*, 18:1.4:1–1.4:43, 2013.

6 Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. Combining Hierarchical and Goal-Directed Speed-up Techniques for Dijkstra's Algorithm. *ACM Journal of Experimental Algorithmics*, 15:2.3:1–2.3:31, 2010.

7 Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest Feasible Paths with Charging Stops for Battery Electric Vehicles. In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'15)*, pages 44:1–44:10. ACM, 2015.

8 Moritz Baum, Julian Dibbelt, Lorenz Hübschle-Schneider, Thomas Pajor, and Dorothea Wagner. Speed-Consumption Tradeoff for Electric Vehicle Route Planning. In *Proceedings of the 14th Workshop on Algorithmic Approaches for Transportation Modeling, Optimization, and Systems (ATMOS'14)*, volume 42 of *OpenAccess Series in Informatics (OASIcs)*, pages 138–151. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/OASIcs.ATMOS.2014.138.

9 Moritz Baum, Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. Energy-Optimal Routes for Electric Vehicles. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'13)*, pages 54–63. ACM, 2013.

10 Luca Bedogni, Luciano Bononi, Marco Di Felice, Alfredo D'Elia, Randolf Mock, Francesco Morandi, Simone Rondelli, Tullio Salmon Cinotti, and Fabio Vergari. An Integrated Simulation Framework to Model Electric Vehicles Operations and Services. *IEEE Transactions on Vehicular Technology*, 65(8):5900–5917, 2016.

11 Marco Blanco, Ralf Borndörfer, Nam-Dung Hoang, Anton Kaier, Adam Schienle, Thomas Schlechte, and Swen Schlobach. Solving Time Dependent Shortest Path Problems on Airway Networks Using Super-Optimal Wind. In *Proceedings of the 16th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS'16)*, volume 54 of *OpenAccess Series in Informatics (OASIcs)*, pages 12:1–12:15. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi:10.4230/OASIcs.ATMOS.2016.12.

12 Karin Brundell-Freij and Eva Ericsson. Influence of Street Characteristics, Driver Category and Car Performance on Urban Driving Patterns. *Transportation Research Part D: Transport and Environment*, 10(3):213–229, 2005.

13 Daniel Delling and Dorothea Wagner. Time-Dependent Route Planning. In *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, pages 207–230. Springer, 2009.

14 Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. User-Constrained Multi-Modal Route Planning. *ACM Journal of Experimental Algorithmics*, 19:3.2:1–3.2:19, 2015.

15 Edsger W. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

16 Jochen Eisner, Stefan Funke, and Sabine Storandt. Optimal Route Planning for Electric Vehicles in Large Networks. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, pages 1108–1113. AAAI Press, 2011.

17 Stephan Erb, Moritz Kobitzsch, and Peter Sanders. Parallel Bi-Objective Shortest Paths Using Weight-Balanced B-Trees with Bulk Updates. In *Proceedings of the 13th International Symposium on Experimental Algorithms (SEA'14)*, volume 8504 of *Lecture Notes in Computer Science*, pages 111–122. Springer, 2014.

18 Chiara Fiori, Kyoungho Ahn, and Hesham A. Rakha. Power-Based Electric Vehicle Energy Consumption Model: Model Development and Validation. *Applied Energy*, 168:257–268, 2016.

19 Carlos Flores, Vicente Milanés, Joshué Pérez, David González, and Fawzi Nashashibi. Optimal Energy Consumption Algorithm Based on Speed Reference Generation for Urban Electric Vehicles. In *Proceedings of the 11th IEEE Intelligent Vehicles Symposium (IV'15)*, pages 730–735. IEEE, 2015.

20 Luca Foschini, John Hershberger, and Subhash Suri. On the Complexity of Time-Dependent Shortest Paths. *Algorithmica*, 68(4):1075–1097, 2014.

21 Stefan Funke and Sabine Storandt. Polynomial-Time Construction of Contraction Hierarchies for Multi-Criteria Objectives. In *Proceedings of the 15th Meeting on Algorithm Engineering & Experiments (ALENEX'13)*, pages 31–54. SIAM, 2013.

22 Robert Geisberger, Moritz Kobitzsch, and Peter Sanders. Route Planning with Flexible Objective Functions. In *Proceedings of the 12th Workshop on Algorithm Engineering & Experiments (ALENEX'10)*, pages 124–137. SIAM, 2010.

23 Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science*, 46(3):388–404, 2012.

24 Andrew V. Goldberg and Chris Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In *Proceedings of the 16th Annual ACM–SIAM Symposium on Discrete Algorithms (SODA'05)*, pages 156–165. SIAM, 2005.

25 Michael T. Goodrich and Paweł Pszona. Two-Phase Bicriterion Search for Finding Fast and Efficient Electric Vehicle Routes. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS'14)*, pages 193–202. ACM, 2014.

26 Gabriel Y. Handler and Israel Zang. A Dual Algorithm for the Constrained Shortest Path Problem. *Networks*, 10(4):293–309, 1980.

27 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

28 Frederik Hartmann and Stefan Funke. Energy-Efficient Routing: Taking Speed into Account. In *Proceedings of the 37th Annual German Conference on Advances in Artificial Intelligence (KI'14)*, volume 8736 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 2014.

29 Stefan Hausberger, Martin Rexeis, Michael Zallinger, and Raphael Luz. Emission Factors from the Model PHEM for the HBEFA Version 3. Technical report I-20/2009, University of Technology, Graz, 2009.

30 James Larminie and John Lowry. *Electric Vehicle Technology Explained, 2nd Edition*. John Wiley & Sons, Ltd., 2012.

31 Mingsong Lv, Nan Guan, Ye Ma, Dong Ji, Erwin Knippel, Xue Liu, and Wang Yi. Speed Planning for Solar-Powered Electric Vehicles. In *Proceedings of the 7th International Conference on Future Energy Systems (e-Energy'16)*, pages 6:1–6:10. ACM, 2016.

32 Enrique Machuca and Lawrence Mandow. Multiobjective Heuristic Search in Road Maps. *Expert Systems with Applications*, 39(7):6435–6445, 2012.

33 Lawrence Mandow and José-Luis Pérez-de-la-Cruz. Multiobjective A* Search with Consistent Heuristics. *Journal of the ACM*, 57(5):27:1–27:24, 2010.

34 Ernesto Q. V. Martins. On a Multicriteria Shortest Path Problem. *European Journal of Operational Research*, 16(2):236–245, 1984.

35 Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient Energy-Optimal Routing for Electric Vehicles. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI'11)*, pages 1402–1407. AAAI Press, 2011.

36 Peter Sanders and Lawrence Mandow. Parallel Label-Setting Multi-Objective Shortest Path Search. In *Proceedings of the 27th IEEE International Parallel and Distributed Processing Symposium (IPDPS'13)*, pages 215–224. IEEE, 2013.

37 Sabine Storandt. Quick and Energy-Efficient Routes: Computing Constrained Shortest Paths for Electric Vehicles. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'12)*, pages 20–25. ACM, 2012.

38 Sabine Storandt. Route Planning for Bicycles – Exact Constrained Shortest Paths Made Practical via Contraction Hierarchy. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*, pages 234–242. AAAI Press, 2012.

39 Tessa Tielert, David Rieger, Hannes Hartenstein, Raphael Luz, and Stefan Hausberger. Can V2X Communication Help Electric Vehicles Save Energy? In *Proceedings of the 12th International Conference on ITS Telecommunications (ITST'12)*, pages 232–237. IEEE, 2012.

40 Chi Tung Tung and Kim Lin Chew. A Multicriteria Pareto-Optimal Path Algorithm. *European Journal of Operational Research*, 62(2):203–209, 1992.

41 Yan Wang, Jianmin Jiang, and Tingting Mu. Context-Aware and Energy-Driven Route Optimization for Fully Electric Vehicles via Crowdsourcing. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1331–1345, 2013.

42 Enjian Yao, Zhiqiang Yang, Yuanyuan Song, and Ting Zuo. Comparison of Electric Vehicle's Energy Consumption Factors for Different Road Types. *Discrete Dynamics in Nature and Society*, 2013.