Shortest Feasible Paths with Charging Stops for Battery Electric Vehicles*

Moritz Baum
Karlsruhe Institute of Technology (KIT)
moritz.baum@kit.edu

Julian Dibbelt
Karlsruhe Institute of Technology (KIT)
dibbelt@kit.edu

Andreas Gemsa
Karlsruhe Institute of Technology (KIT)
gemsa@kit.edu

Dorothea Wagner
Karlsruhe Institute of Technology (KIT)
dorothea.wagner@kit.edu

Tobias Zündorf
Karlsruhe Institute of Technology (KIT)
zuendorf@kit.edu

ABSTRACT

We study the problem of minimizing *overall* trip time for battery electric vehicles (EVs) in road networks. As battery capacity is limited, stops at charging stations may be inevitable. Careful route planning is crucial, since charging stations are scarce and recharging is time-consuming. We extend the Constrained Shortest Path (CSP) problem for EVs with *realistic* models of charging stops, including varying charging power and battery swapping stations. While the resulting problem is $\mathcal{NP}$-hard, we propose a combination of algorithmic techniques to achieve good performance in practice. Extensive experimental evaluation shows that our approach (CHArge) enables computation of *optimal* solutions on realistic inputs, even of continental scale. Finally, we investigate heuristic variants of CHArge that derive high-quality routes in well below a second on sensible instances.

Categories and Subject Descriptors

G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms, network problems*; G.2.3 [**Discrete Mathematics**]: Applications

General Terms

Algorithms, Experimentation, Measurement, Performance

Keywords

route planning, road networks, speedup technique, shortest paths, electric vehicles, energy consumption, charging station

*Supported by EU FP7, grant no. 609026 (MOVESMART).

SIGSPATIAL'15, November 3–6, 2015, Bellevue, WA, USA

ACM 978-1-4503-3967-4/15/11 ...$15.00.
DOI: http://dx.doi.org/10.1145/2820783.2820826

1. INTRODUCTION

Electromobility promises independence of fossil fuels, zero (local) emissions, and higher energy-efficiency, especially for city traffic. Yet, most of the significant algorithmic progress on route planning in road networks [2] has focused on conventional combustion-engine cars. EVs, however, have limited driving range, charging stations are still much rarer than gas stations, and recharging is time-consuming. Thus, routes can become infeasible (the battery runs empty), and fast routes may be less favorable when accounting for recharging time.

Related work. Classic route planning approaches apply Dijkstra's algorithm [10] to a graph representation of the transportation network. For faster queries, *speedup techniques* [2] have been proposed, with different benefits in terms of preprocessing time and space, query speed, and simplicity. A* Search [21] uses vertex potentials to guide the search towards the target. A successful variant, ALT (A*, Landmarks, Triangle inequality) [18], obtains good potentials from precomputed distances. Contraction Hierarchies (CH) [16], on the other hand, iteratively *contract* vertices during preprocessing, maintaining distances between all remaining vertices by adding *shortcut* arcs where necessary. The CH query is then bidirectional, starting from source and target, and proceeds only from less important to more important vertices. Combining both techniques, Core-ALT [3] contracts all but the most important vertices, performing ALT on the remaining *core* graph. This approach can also be extended to more complex scenarios, such as edge constraints (modeling, e. g., maximum allowed vehicle height or weight) [15]. More recently, techniques (including variants of CH and ALT) were introduced that allow an additional *customization* after preprocessing, to account for dynamic or user-dependent metrics [6, 9, 11]. Also, approaches towards extended scenarios exist, such as shortest via paths [1, 8] or batched shortest paths [7]. Here, a common approach is to make use of a (relatively fast) *target selection* phase, precomputing distances to relevant points of interest to allow for faster queries. However, the techniques mentioned above were only evaluated for single-criterion search (where distance between two vertices is always a unique scalar value). For multi-criteria scenarios, on the other hand, problem complexity and solution sizes increase significantly, and practical approaches are only

known for basic problem variants [13, 14]. See [2] for a more complete overview of techniques and combinations.

For EVs, the battery must not deplete, but energy is recuperated when braking (e. g., going downhill), up to the maximum capacity. These *battery constraints* must be checked during route computation. This can be done implicitly, by weighting each road segment with a *consumption profile*, mapping current *state of charge* (SoC) to actual consumption wrt. constraints [5, 12]. Still, both preprocessing and queries are slower, compared to the standard shortest path problem, if battery constraints have to be obeyed [5, 12, 30]. Several natural problem formulations specific to EV routing exist. Some optimize energy consumption [5, 12, 28], often, however, resulting in routes resorting to minor (i. e., slow) roads to save energy. *Constrained Shortest Path (CSP)* [20] formulations ask to find the most energy-efficient route without exceeding a certain driving time — or finding the fastest route that does not violate battery constraints [30]. Similarly, one can compute all trade-offs between driving time and energy consumption (or a subset for faster queries, but dropping optimality [4]). Another approach separates queries into two phases, optimizing driving time and energy consumption, respectively [19]. Techniques presented in [4, 19] allow for reasonably fast queries (even without preprocssing), but results are not optimal in general (i. e., there might be paths with lower trip time that respect battery constraints).

Without recharging, large parts of the road network are simply not reachable by an EV, rendering long distance trips impossible. (For conventional cars, broad availability of gas stations and short refuel duration allow to neglect this in route optimization.) Charging stations have been considered by [19, 29, 30] under the simplifying assumption that using a charging station always results in a fully-recharged battery, and that the charging process takes constant time (independent of the initial SoC). As such, feasible paths between stations are independent of source and target, hence easily precomputed. However, this assumption only holds for battery swapping stations, which are still an unproven technology and business model. For regular charging stations, charging time depends on the desired SoC. Moreover, while nearly linear for low SoC, the charging rate decreases when approaching the battery limit. Thus it can be reasonable to only charge up to a fraction of the battery's limit. In [31], this behavior is modeled by combining a linear with an exponential function for high SoC. In [25], recharging above 80 % SoC is suppressed altogether. However, neither approach was shown to scale to road networks of realistic size. Also, while omitting the possibility of charging beyond 80% might be appropriate for regions well covered with charging stations, it drastically decreases reachability in regions with only few charging stations, where recharging to a full battery can become inevitable.

Our Contribution. In this work, we extend the CSP problem to planning routes that, while respecting battery constraints, minimize overall trip time, including time spent at charging stations. Our solution handles all types of stations accurately: battery swapping stations, regular charging stations with various charging powers, as well as superchargers. In particular, charging times are *not* independent of the remaining SoC when arriving at a charging station. Additionally, the charging process can be interrupted as soon as further charging would increase the arrival time at the target. Since the resulting problem is $\mathcal{NP}$-hard, we do not guarantee polynomial running times. However, carefully incorporating recharging in speedup techniques, enables us to solve the problem optimally and within seconds, even for continental road networks. For even faster queries, we propose heuristic approaches that offer high (empirical) quality. Extensive experiments on detailed and realistic data show that our approach, while designed to solve a more complex problem, clearly outperforms and broadens the state-of-the-art.

Outline. Section 2 sets necessary notation. Section 3 specifies the problem formulation, while Section 4 presents our basic approach. In Section 5, we propose speedup techniques. Section 6 introduces heuristics, which drop correctness for faster queries. Section 7 experimentally evaluates all approaches. We close with final remarks in Section 8.

2. PRELIMINARIES

We consider *directed, weighted* graphs $G = (V, A)$, with two arc weight functions $d\colon A \to \mathbb{R}_{\geq 0}$ and $c\colon A \to \mathbb{R}$, representing driving time and energy consumption on an arc, respectively. Note that consumption can be negative due to recuperation (though cycles with negative consumption are physically ruled out). An *s–t-path* in a graph G is defined as a sequence $P = [s = v_1, v_2 \ldots, v_k = t]$ of vertices, such that $(v_i, v_{i+1}) \in A$ for $1 \leq i \leq k-1$. If $s = t$, we call P a *cycle*. Given two paths $P = [v_1, \ldots, v_i]$ and $Q = [v_i, \ldots, v_k]$, we denote by $P \circ Q := [v_1, \ldots, v_i, \ldots, v_k]$ their concatenation. The *driving time* on a path P is $d(P) = \sum_{i=1}^{k-1} d(v_i, v_{i+1})$. For energy consumption this is more involved: the battery has a limited capacity M (which can neither be exceeded nor drop below zero), so we must take *battery constraints* into account. To reflect battery constraints on an s–t-path P, we define its *consumption profile* $f_P\colon [0, M] \to [-M, M] \cup \{\infty\}$ that determines the actual consumption depending on the SoC β_s at s, so the SoC at t is $\beta_t = \beta_s - f_P(\beta_s)$; see Figure 1 for an example. We use the value ∞ to indicate that the SoC at s is not sufficient to reach t, i. e., P is not *feasible* for the corresponding SoC at s. As shown in [12], f_P can be represented using only three values, namely, the minimum SoC $\text{in}_P \in [0, M]$ required to traverse P, its energy consumption $\text{cost}_P \in [-M, M]$ (which can be less than in_P due to recuperation), and the maximum possible SoC after traversing P, denoted $\text{out}_P \in [0, M]$. We then have

$$f_P(\beta) := \begin{cases} \infty & \text{if } \beta < \text{in}_P, \\ \beta - \text{out}_P & \text{if } \beta - \text{cost}_P > \text{out}_P, \\ \text{cost}_P & \text{else.} \end{cases}$$

For an arc $a \in A$, f_a is given by $\text{cost}_a := c(a)$, $\text{in}_a := \max\{0, c(a)\}$, and $\text{out}_a := \min\{M, M - c(a)\}$. For two consumption profiles f_P and f_Q of paths P and Q, we obtain the *linked* profile $f_{P \circ Q}$ by setting

$$\begin{aligned} \text{in}_{P\circ Q} &:= \max\{\text{in}_P, \text{cost}_P + \text{in}_Q\} \\ \text{out}_{P\circ Q} &:= \min\{\text{out}_Q, \text{out}_P - \text{cost}_Q\} \\ \text{cost}_{P\circ Q} &:= \max\{\text{cost}_P + \text{cost}_Q, \text{in}_P - \text{out}_Q\}. \end{aligned}$$

An example of two consumption profiles as well as the result of linking them is shown in Figure 1.

Given a graph, two (nonnegative) functions (representing weight and consumption) on its arcs, and vertices s and t, the Constrained Shortest Path Problem asks for a path of

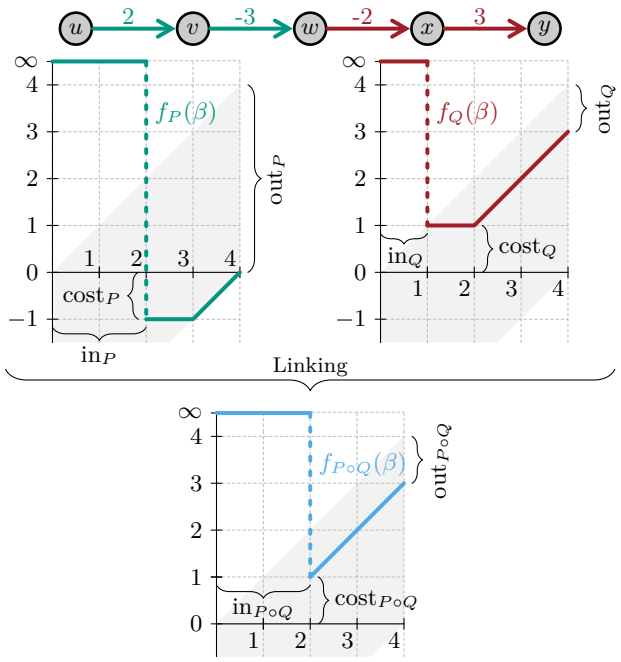

Figure 1: Two consumption profiles f_P, f_Q (for paths $P = [u, v, w]$ and $Q = [w, x, y]$) and the resulting profile $f_{P\circ Q}$ after linking them. The battery capacity is $M = 4$. The first profile f_P is represented by $\text{cost}_P = -1$, $\text{in}_P = 2$ and $\text{out}_P = 4$, the profile f_Q is given by $\text{cost}_Q = 1$, $\text{in}_Q = 1$ and $\text{out}_Q = 1$. Linking these profiles yields $\text{cost}_{P\circ Q} = 1$, $\text{in}_{P\circ Q} = 2$ and $\text{out}_{P\circ Q} = 1$. (Observe that $\text{cost}_{P\circ Q}$ is greater than the sum $\text{cost}_P + \text{cost}_Q = 0$.)

minimum weight such that its consumption does not exceed a certain bound. While being $\mathcal{NP}$-hard [20], CSP can be solved using a (exponential-time) bicriteria variant [26] of Dijkstra's algorithm [10]. In our scenario, this algorithm maintains a *label set* $L(\cdot)$ of *labels* (tuples of driving time and SoC) for each $v \in V$. A label *(Pareto) dominates* another label of the same vertex, if it is better in one criterion and not worse in the other. Initially, all label sets are empty, except for the label $(0, \beta_s)$ at the source s, which is also added to a priority queue. In each step, the algorithm *settles* the minimum label ℓ of the queue. This is done by extracting the label $\ell = (\tau, \beta)$ (associated to some vertex u), and scanning all arcs (u, v) outgoing from u. If the new label $\ell' := (\tau + d(u, v), \beta - c(u, v))$ is not dominated by any label in $L(v)$, it is added to $L(v)$ (removing labels dominated by ℓ' from $L(v)$) and the queue. Using driving time as key in the priority queue (breaking ties by SoC), the algorithm is *label setting* (extracted labels are never dominated). An optimal (constrained) path is found once a label (with nonnegative SoC) at t is extracted. Battery constraints can be incorporated by on-the-fly checks during the algorithm [4].

A *potential function* $\pi\colon V \to \mathbb{R}$ is *consistent* (wrt. driving time) if $d((u, v)) - \pi(u) + \pi(v) \geq 0$ for all $(u, v) \in A$. The A* algorithm [21] adds the potential of a vertex to the keys of its labels, changing the order in which they are extracted.

In CH [16], vertices are iteratively contracted during preprocessing (according to a given vertex ordering), while introducing *shortcuts* between their neighbors to maintain distances (wrt. d and c). To avoid unnecessary shortcuts, *witness searches* are run between neighbors to identify existing paths that dominate a shortcut candidate. Adding shortcuts may lead to (nondominated) multi-arcs. A CH query runs bidirectional, scanning only upward arcs (wrt. the ordering) in the forward search, and downward arcs in the backward search.

3. PROBLEM STATEMENT

In this work, we consider stops at charging stations to recharge the battery (while spending charging time). In our model, a subset $S \subseteq V$ of the vertices represents charging stations. Each vertex $v \in S$ has a designated *charging function* $\text{cf}_v\colon [0, M] \times \mathbb{R}_{\geq 0} \to [0, M]$, mapping initial SoC and charging time to the resulting SoC. We presume that charging functions are concave (i. e., charging speed decreases as the battery's SoC increases) and monotonically increasing wrt. charging time (i. e., charging for a longer time never decreases the SoC). Additionally, we demand that for arbitrary charging times $\tau_1, \tau_2 \in \mathbb{R}_{\geq 0}$ and SoC values $\beta \in [0, M]$ the *shifting property* $\text{cf}_v(\text{cf}_v(\beta, \tau_1), \tau_2) = \text{cf}_v(\beta, \tau_1 + \tau_2)$ holds. These conditions are met by realistic physical models of charging stations [32]. Moreover, exploiting the shifting property, it is possible to represent the (bivariate) charging function cf_v using a univariate function $\widetilde{\text{cf}}_v\colon \mathbb{R}_{\geq 0} \to [0, M]$ with $\widetilde{\text{cf}}_v(\tau) := \text{cf}_v(0, \tau)$. Let $\text{cf}_v^{-1}(\beta_1, \beta_2) := \widetilde{\text{cf}}_v^{-1}(\beta_2) - \widetilde{\text{cf}}_v^{-1}(\beta_1)$ denote the time to charge the battery from β_1 to β_2, then we have $\text{cf}_v(\beta, \tau) = \widetilde{\text{cf}}_v(\tau + \text{cf}_v^{-1}(0, \beta))$.

The notion of charging functions is flexible enough to represent realistic charging stations, including, e. g., swapping stations ($\text{cf}(\tau, \beta) = M$ for all $\tau \in \mathbb{R}_{\geq 0}, \beta \in [0, M]$). Finally, we assign to every $v \in S$ a constant *arrangement time* $\tau_a(v)$ that is spent when charging at v. Thereby, we model time overhead at a charging station (e. g., parking the car or swapping the battery). For the sake of simplicity (and motivated by data input used in our experimental evaluation), we assume that charging functions are not only concave and increasing — but also *piecewise linear*. However, this is not a necessary condition for correctness of our approaches.

We consider the following objective: for given $s, t \in V$ and an initial SoC $\beta_s \in [0, M]$, find a feasible s–t-path that minimizes the *trip time* (the sum of driving time and total charging time at charging stations). Note that if the input graph contains neither charging stations ($S = \emptyset$) nor arcs with negative consumption values, we have an instance of CSP, hence the problem is $\mathcal{NP}$-hard.

4. BASIC APPROACH

Since charging functions are continuous, there is no straightforward way to apply the bicriteria algorithm described in Section 2 to our setting (as it might require an infinite number of Pareto optimal labels after settling a charging station). Hence, we generalize the bicriteria approach to our setting. The charging function propagating (CFP) algorithm extends labels to represent (infinite, continuous) sets of nondominated solutions. The core idea is that a label now represents all possible trade-offs between charging time and resulting SoC induced by the last visited charging station (if it exists).

Our algorithm is initialized with a single label $(0, \beta_s, \bot, \bot)$ at the source s, and proceeds as the bicriteria algorithm described in Section 2. Additionally, when extracting a label at some vertex u, we check whether $u \in S$. If this is the case, we create new labels to account for possible recharging. There

are, in general, infinitely many feasible, nondominated pairs of charging time and resulting SoC (one for each reasonable charging time $\tau_c \in [0, \mathrm{cf}_u^{-1}(\beta, M)]$). We implicitly represent these pairs in one label by storing the charging station u in the label. Observe that this no longer allows us to apply battery constraints on-the-fly: for vertices v visited after u, labels have no fixed SoC (it depends on how much energy was charged at u). Hence, we compute the consumption profile $f_{[u,\dots,v]}$ of the subpath from u to v. Therefore, a label $\ell = (\tau_\mathrm{t}, \beta_u, u, f_{[u,\dots,v]})$ at vertex v consists of the trip time τ_t of the path from s to v (including charging time on every previous charging station except u), the SoC β_u when reaching u, the last charging station u, and the consumption profile $f_{[u,\dots,v]}$ of the subpath from u to v.

When reaching a new charging station v, we have to replace the last station u of the current label ℓ (unless no station was visited before). We do so by fixing the charging time at u. Then we can evaluate $f_{[u,\dots,v]}$, and thereby determine the SoC β_v at v for the label ℓ. However, we still face the problem that there are (in general) infinitely many possible charging times (at u). Theorem 1 shows that if charging at v, we only have to consider a small (finite) number of charging times at u. Thus, spawning a small number of new labels (each fixing a certain charging time at u and setting the last charging station to v) suffices to represent all nondominated solutions. While Theorem 1 considers only piecewise linear functions, a similar property can be shown for general (convex, increasing) functions.

In order to prove Theorem 1, we define the *SoC-function* b_ℓ of a label ℓ, to represent all feasible pairs of trip time and SoC associated with the label $\ell = (\tau_\mathrm{t}, \beta_u, u, f_{[u,\dots,v]})$. Let $\tau' := \tau_\mathrm{t} + \tau_a(u)$, then the SoC-function b_ℓ is defined as

$$b_\ell(\tau) := \mathrm{cf}_u(\beta_u, \tau - \tau') - f_{[u,\dots,v]}(\mathrm{cf}_u(\beta_u, \tau - \tau'))$$

for all $\tau \geq \tau'$, else $b_\ell(\tau) := -\infty$; see Figure 2 for an example. The definition of SoC-functions reflects the interpretation of our labels, which represent all trade-offs between charging time and resulting SoC induced by the charging function cf_u of the last station. Note that, while charging functions can have arbitrary complexity, we propagate them using labels of constant size. Assume that v is another charging station, then we have to set a charging time τ_c at u (so that a new label with v as its last visited charging station can be created). For a given SoC-function b_ℓ at a charging station $v \in S$, we define $b_\ell^\tau := b_{\ell'}$ with $\ell' := (\tau, b_\ell(\tau), v, 0)$. The function b_ℓ^τ corresponds to picking some charging time τ_c for charging at u, such that the trip time from s to v is $\tau = \tau_\mathrm{t} + \tau_a + \tau_c$, and v is reached with a SoC of $b_\ell(\tau)$. We now prove Theorem 1, stating that we require new SoC-functions b_ℓ^τ only for a small number of values τ.

Theorem 1. *For a vertex $v \in S$ and a label ℓ at v, let the associated SoC-function b_ℓ be represented by the sequence $[(\tau_1, \beta_1), \dots, (\tau_k, \beta_k)]$ of breakpoints (i. e., $\tau_i < \tau_j$ for $i < j$, $b_\ell(x) = -\infty$ for $\tau < \tau_1$ and $b_\ell(\tau) = \beta_k$ for $\tau > \tau_k$). For every $\tau \geq \tau_1$, there exists an $i \in \{1, \dots, k\}$ such that $b_\ell^{\tau_i}$ dominates b_ℓ^τ (i. e., $b_\ell^{\tau_i}(x) \geq b_\ell^\tau(x)$ for all $x \geq 0$).*

Proof. We prove this theorem in two steps. First, we show that $b_\ell^{\tau_k}$ is greater (or equal to) b_ℓ^τ for the special case of $\tau \geq \tau_k$. Afterwards, we prove the claim for the case of τ being between two breakpoints, i. e., $\tau_1 \leq \tau \leq \tau_k$. In what follows, let $\tau' := \tau + \tau_a(v)$, and let $\ell = (\tau_\mathrm{t}, \beta_u, u, f_{[u,\dots,v]})$.

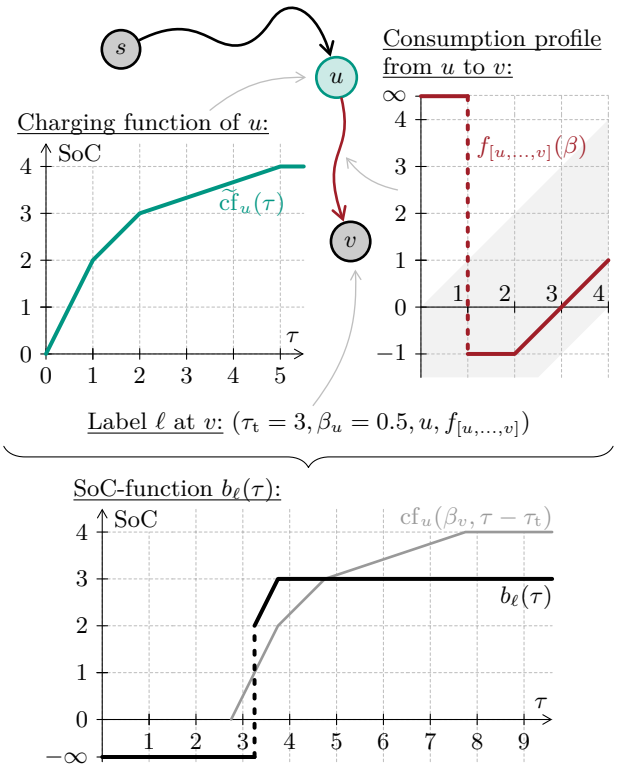

Figure 2: Construction of the SoC-function b_ℓ for a given label $\ell = (\tau_\mathrm{t}, \beta_u, u, f_{[u,\dots,v]})$, representing a path in the graph from s to v. The last charging station u was reached with a SoC of $\beta_u = 0.5$. The function $\widetilde{\mathrm{cf}}_u$ is shown on the left, its arrangement time is $\tau_a(u) = 0$. Consumption on the subpath from u to v is given by the profile $f_{[u,\dots,v]}$ on the right. Without recharging at u, it takes $\tau_\mathrm{t} = 3$ time to reach v. The function $\mathrm{cf}_u(\beta_u, \tau - \tau_\mathrm{t})$ (in gray; bottom) reflects pairs of trip time and SoC when charging at u, but ignores consumption on the u–v-subpath. To obtain $\mathrm{cf}_u(\beta_u, \tau - \tau_\mathrm{t})$, we shift $\widetilde{\mathrm{cf}}_u$ by $\tau_\mathrm{t} = 3$ to the right and by $\mathrm{cf}_u^{-1}(0, \beta_u = 0.5) = 0.25$ to the left (see Section 3). We subtract the consumption $f_{[u,\dots,v]}(\mathrm{cf}_u(0.5, \tau - 3))$ on the u–v-path, and obtain $b(\tau)$ (in black; bottom). We have $b(\tau) = -\infty$ for $\tau < 3.25$ (the minimum charging time at u is 0.25, as a SoC of 1 is required to reach v) and $b(\tau) = 3$ for $\tau \geq 3.75$ (charging beyond SoC of 2 never pays off, as it wastes energy from recuperation).

Case 1: $\tau \geq \tau_k$. We have to show that $b_\ell^{\tau_k}(x) \geq b_\ell^\tau(x)$ holds for all $x \geq 0$. In the case of $x < \tau'$ we have $b_\ell^\tau(x) = -\infty$, hence $b_\ell^{\tau_k}(x) \geq b_\ell^\tau(x)$ holds. For $x \geq \tau'$, let $\tau_k' := \tau_k + \tau_a(v)$, then we have

$$\begin{aligned} b_\ell^{\tau_k}(x) &= \mathrm{cf}_v(b_\ell(\tau_k), x - \tau_k') - f_{[u,\dots,v]}(\mathrm{cf}_u(b_\ell(\tau_k), x - \tau_k')) \\ &= \mathrm{cf}_v(b_\ell(\tau), x - \tau_k') - f_{[u,\dots,v]}(\mathrm{cf}_u(b_\ell(\tau), x - \tau_k')) \\ &= b_\ell^\tau(x + \tau - \tau_k) \geq b_\ell^\tau(x). \end{aligned}$$

First, we can replace τ_k with τ in the definition of $b_\ell^{\tau_k}(\cdot)$, since $b_\ell(\cdot)$ is constant for values greater or equal to τ_k. This leaves us with $b_\ell^\tau(\cdot)$, only shifted by $\tau - \tau_k$, which (according to our assumption) is greater or equal to 0. As all involved

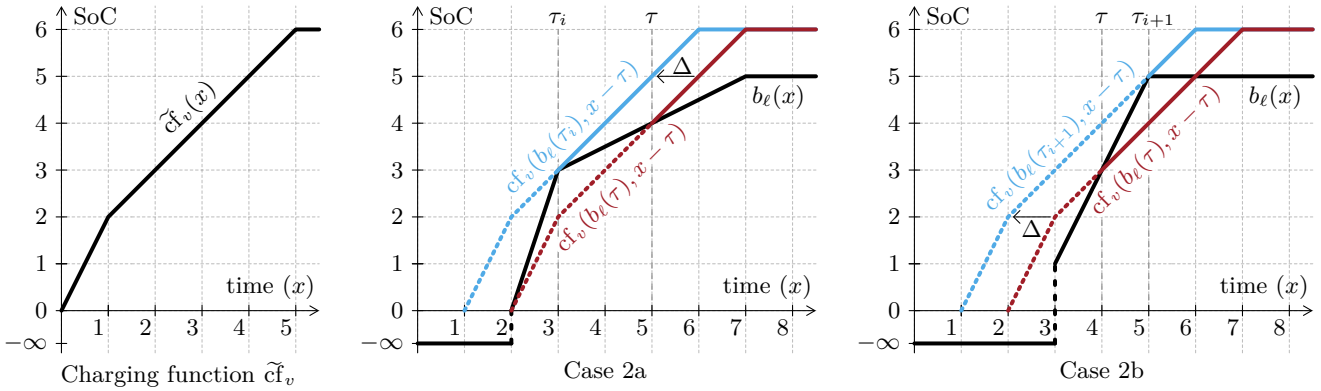

Figure 3: Examples for cases 2a and 2b. The charging function at v is given by the sequence $[(0,0),(1,2),(5,6)]$ in both cases ($\tau_a(v)=0$). For case 2a, the SoC-function b_ℓ is defined by $[(2,0),(3,3),(7,5)]$, so the slope of b_ℓ at $x=\tau$ is lower than the slope of $\mathrm{cf}_v(b_\ell(\tau),x)$ at $x=\tau$. For case 2b, the SoC-function b_ℓ is given by $[(3,1),(5,5)]$, thus the slope of b_ℓ at $x=\tau$ is greater than the slope of $\mathrm{cf}_v(b_\ell(\tau),x)$ at $x=\tau$. In both cases the charging function can be shifted to the left by Δ, such that it intersects b_ℓ at a breakpoint and the initial function is dominated by the shifted one. (Note that a function dominates the area beneath it).

charging functions are monotonically increasing, this shift can only increase the value of the function b_ℓ^τ. Thus, $b_\ell^{\tau_k}$ dominates b_ℓ^τ for $\tau \geq \tau_k$.

Case 2: $\tau < \tau_k$. In this case, there is an $i \in \{1,\dots,k-1\}$, such that $\tau_i \leq \tau < \tau_{i+1}$ holds (recall that $\tau \geq \tau_1$ holds by assumption). We now show that, depending on the slopes of b_ℓ and cf_v at τ, either $b_\ell^{\tau_i}$ or $b_\ell^{\tau_{i+1}}$ dominates b_ℓ^τ. To formally define the *slope* of some piecewise linear function g, let $(\partial g(x')/\partial x')(x)$ denote its *right derivative* at some $x \in \mathbb{R}_{\geq 0}$ (hence, slope is well-defined at breakpoints).

Case 2a: We first consider the case of

$$\frac{\partial b_\ell(x)}{\partial x}(\tau) = \frac{b_{i+1}-b_i}{\tau_{i+1}-\tau_i} < \frac{\partial\, \mathrm{cf}_v(b_\ell(\tau), x-\tau')}{\partial x}(\tau).$$

Intuitively, this means that the charging station at v provides a better charging rate than the station at u (if the battery is charged at u such that v is reached at time τ with SoC $b(\tau)$). Hence, leaving u earlier to charge more energy at v (to benefit from the higher charging rate) pays off. This implies that $b_\ell^{\tau_i}$ dominates b_ℓ^τ, which we now prove formally, by exploiting that b_ℓ is piecewise linear and cf_v is concave.

For all $x < \tau'$ we know that $b_\ell^\tau(x) = -\infty$ (and thus is dominated by $b_\ell^{\tau_k}(x)$). For $x \geq \tau'$, we can make use of the shifting property to obtain some value Δ such that

$$\mathrm{cf}_v(b_\ell(\tau_i), x-\tau_i') = \mathrm{cf}_v(b_\ell(\tau), x+\Delta-\tau'),$$

where $\tau_i' := \tau_i + \tau_a(v)$. In other words, we change the point in time at which we stop charging at u to spend more time charging at v from τ to τ_i, which is equivalent to shifting the charging function cf_u on the x-axis; see Figure 3 for an example. Recall that $\mathrm{cf}_v^{-1}(\beta_1,\beta_2)$ is the time required to charge the battery from β_1 to β_2 (see Section 3). We obtain

$$\begin{aligned}
&\mathrm{cf}_v\left(b_\ell(\tau_i), x-\tau_i'\right)\\
&\quad= \mathrm{cf}_v\left(0, x-\tau_i' + \mathrm{cf}_v^{-1}(0, b_\ell(\tau_i))\right)\\
&\quad= \mathrm{cf}_v\left(b_\ell(\tau), x-\tau_i' + \mathrm{cf}_v^{-1}(0, b_\ell(\tau_i)) - \mathrm{cf}_v^{-1}(0, b_\ell(\tau))\right)\\
&\quad= \mathrm{cf}_v\left(b_\ell(\tau), x-\tau_i' - \mathrm{cf}_v^{-1}(b_\ell(\tau_i), b_\ell(\tau))\right)\\
&\quad= \mathrm{cf}_v\left(b_\ell(\tau), x-\tau' + \tau' - \tau_i' - \mathrm{cf}_v^{-1}(b_\ell(\tau_i), b_\ell(\tau))\right),
\end{aligned}$$

and therefore, $\Delta = \tau' - \tau_i' - \mathrm{cf}_v^{-1}(b(\tau_i), b(\tau))$. Furthermore, $\Delta \geq 0$ holds, since $\tau' - \tau_i' = \tau - \tau_i \geq \mathrm{cf}_v^{-1}(b(\tau_i), b(\tau))$. Here, $\tau - \tau_i$ is the time needed to charge from $b_\ell(\tau_i)$ to $b_\ell(\tau)$ (using cf_u at u), and $\mathrm{cf}_v^{-1}(b_\ell(\tau_i), b_\ell(\tau))$ is the time needed to charge the same amount at the station v. We know that the slope $\delta_{b_\ell}(\tau_i) = \delta_{b_\ell}(\tau)$ (at the charging station u) is (strictly) less than the slope $\delta_{\mathrm{cf}_v}(\tau_i) \geq \delta_{\mathrm{cf}_v}(\tau)$ at the charging station v. Therefore, the station v requires less time to charge from $b_\ell(\tau_i)$ to $b_\ell(\tau)$. Given that $\Delta \geq 0$, for $x \geq \tau'$ we have

$$\begin{aligned}
b_\ell^{\tau_i}(x) &= \mathrm{cf}_v(b_\ell(\tau_i), x-\tau_i') - f_{[u,\dots,v]}(\mathrm{cf}_u(b_\ell(\tau_i), x-\tau_i'))\\
&= \mathrm{cf}_v(b_\ell(\tau), x+\Delta-\tau') - f_{[u,\dots,v]}(\mathrm{cf}_u(b_\ell(\tau), x+\Delta-\tau'))\\
&= b_\ell^\tau(x+\Delta) \geq b_\ell^\tau(x).
\end{aligned}$$

Case 2b: The second case we have to address is

$$\frac{\partial b_\ell(x)}{\partial x}(\tau) = \frac{b_{i+1}-b_i}{\tau_{i+1}-\tau_i} \geq \frac{\partial\, \mathrm{cf}_v(b_\ell(\tau), x-\tau')}{\partial x}(\tau).$$

An example for this case is shown in Figure 3b. In this case, the charging station at u offers a more (or equally) favorable charging rate, thus it pays off to spend more time charging at u. Proceeding along the lines of case 2a, we obtain a value $\Delta = \mathrm{cf}_v^{-1}(b(\tau), b(\tau_{i+1})) - (\tau_{i+1}' - \tau')$ such that

$$\mathrm{cf}_v(b_\ell(\tau_{i+1}), x-\tau_{i+1}') = \mathrm{cf}_v(b_\ell(\tau), x+\Delta-\tau'),$$

The value Δ is the difference between the time to charge from $b_\ell(\tau)$ to $b_\ell(\tau_{i+1})$ at v and the time to charge the same amount at u. Since u offers a charging speed at least as high as the one at v, this difference, and therefore Δ, is again not negative. Thus, we can show (similar to case 2a) that $b_\ell^{\tau_{i+1}}(x) \geq b_\ell^\tau(x)$ for all $x \geq \tau'$ (the case $x < \tau'$ is again trivial). This completes the proof. □

Consequently, we spawn one new label, representing the charging function cf_v at v, for each breakpoint of b_ℓ. Note that the original label ℓ is not thrown away (accounting for the possibility of not using the charging station at v at all).

After checking if the current vertex v is a charging station (possibly creating and enqueuing new labels at v), we scan all outgoing arcs (v,w). Let the current label be $\ell = (\tau_\mathrm{t}, \beta, u, f_{[u,\dots,v]})$, then scanning the arc (v,w) creates the new label $\ell' = (\tau_\mathrm{t} + d(v,w), \beta, u, f_{[u,\dots,v]\circ(v,w)})$, which can

be computed in constant time (and independent of the charging function complexity). The new label ℓ' is then added to the label set at vertex w. Note that we perform no dominance checks at this point. Instead, we split label sets $L(\cdot)$ into two sets $L_{\text{set}}(\cdot)$ and $L_{\text{uns}}(\cdot)$ containing settled (i. e., extracted) and unsettled labels, respectively. Sets $L_{\text{uns}}(\cdot)$ are organized as min-heaps, allowing efficient extraction of the next unsettled vertex. We maintain the invariant that for each $v \in V$, the minimum label ℓ (wrt. its key, defined as the minimum feasible trip time of b_ℓ) in $L_{\text{uns}}(v)$ is not dominated by any label in $L_{\text{set}}(v)$. Every time an element of the heap is extracted (or added), we check whether the new minimum is dominated by a label in $L_{\text{set}}(v)$, and remove it in this case (as it cannot lead to an optimal solution). Using heaps for unsettled labels, we avoid unnecessary dominance checks. To determine whether a label ℓ dominates another label ℓ', we check whether $b_\ell(\tau) \geq b_{\ell'}(\tau)$ for all $\tau \geq 0$ (which requires a linear sweep over the breakpoints of both SoC-functions).

When extracting a label at the target vertex t, we pick the least charging time at the last station such that t can be reached, and the algorithm terminates. Correctness of CFP follows immediately from Theorem 1. We can also handle related query types (e. g., arrive at t with at least $x\%$ SoC). For path unpacking, we add two pointers to each label, storing its parent vertex and parent label (for $v \in S$, v can be its own parent). Two consecutive identical parents imply the use of a charging station, the according charging time is the difference between the trip times of both labels.

5. SPEEDUP TECHNIQUES

We present techniques based on A* Search, CH, and a combination of them, to reduce the running times of the basic approach, CFP, introduced in Section 4.

A Search.* We describe potential functions that are based on backward searches from the target vertex t, providing lower bounds on the trip time from any vertex to t. Clearly, we can obtain a consistent potential π_d by running a single-criterion (backward) Dijkstra from t using d as metric, which yields for each $v \in V$ its minimum (unconstrained) driving time to t. However, we can do better, exploiting that the trip time from v to t depends on the SoC β at v. (Observe that both the charging time as well as the route, and hence, the driving time of the optimal v–t-path can change with varying values of β.) Therefore, we propose a potential function $\pi\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$, such that $\pi(v, \beta)$ yields a lower bound on the trip time from v to t if the SoC at v is β. To maintain correctness of our approach, we generalize the notion of consistency: a potential function $\pi\colon V \times [0, M] \to \mathbb{R}$ is consistent if $d(u, v) - \pi(u, \beta) + \pi(v, \beta - f_{(u,v)}(\beta)) \geq 0$ holds for all $(u, v) \in A$ and $\beta \in [0, M]$. Moreover, for the SoC-function b_ℓ represented by a label $\ell = (\tau_{\text{t}}, \beta, u, f_{[u,\dots,v]})$ at v, let $k(\tau, b_\ell(\tau)) := \tau + \pi(v, b_\ell(\tau))$ denote the key (used in the priority queue) of some point $(\tau, b_\ell(\tau))$ of the SoC-function. Then the key $k(\ell)$ of ℓ is *consistent* if $k(\ell) \leq k(\tau, b_\ell(\tau))$ for all $\tau \geq 0$. If label keys as well as the potential function are consistent (as is the case for the basic approach from Section 4), keys of labels extracted from the priority queue are increasing (note that labels spawned at charging stations never lead to decreasing keys). Thus, the algorithm is label setting (implying correctness).

For our first potential function, let $\text{cf}_{\max}$ denote the maximum slope of any charging function in S (i. e., the maximum charging speed available). At swapping stations, we incorporate arrangement time to obtain finite slopes. We define a new weight function $\omega\colon A \to \mathbb{R}$, $\omega(a) := d(a) + (c(a)/\,\text{cf}_{\max})$. This function adds to the driving time of every arc a (possibly negative) lower bound on the time required for charging the energy consumed on the arc. Before an s–t query, we perform three (backward) single-criterion Dijkstra runs from t to obtain, for all $v \in V$, the distances $\text{dist}_d(v, t)$, $\text{dist}_c(v, t)$, and $\text{dist}_\omega(v, t)$ from v to t, wrt. the edge weights d, c, and ω, respectively. Note that computation of $\text{dist}_c(v, t)$ and $\text{dist}_\omega(v, t)$ is label correcting, due to negative weights. (We omit potential shifting [24] because the effect on overall running time is negligible.) Using these distances, we define a potential function π_ω by setting $\pi_\omega(v, \beta) := \text{dist}_d(v, t)$ if $\beta \geq \text{dist}_c(v, t)$, and $\pi_\omega(v, \beta) := \text{dist}_\omega(v, t) - (\beta/\,\text{cf}_{\max})$ otherwise. The resulting potential function π_ω provides a lower bound on the remaining trip time on any path to t. It is easy to see that the potential is also consistent. Regarding consistent label keys, observe that both $\tau + \text{dist}_d(v, t)$ and $\tau + \text{dist}_\omega(v, t) - (b_\ell(\tau)/\,\text{cf}_{\max})$ are increasing in τ. We obtain the consistent key of a label ℓ by computing the value $k(\tau, b_\ell(\tau)) = \tau + \pi_\omega(v, b_\ell(\tau))$ for the first feasible point $(\tau_1, b_\ell(\tau_1))$ of b_ℓ and for the first point $(\tau_2, b_\ell(\tau_2))$ at which $b_\ell(\tau_2) \geq \text{dist}_c(v, t)$, if it exists. The minimum of both these values is used as consistent key.

While the potential π_ω incorporates SoC, it may be too conservative in that it presumes recharging is possible at any time and at the best charging rate. We attempt to be more precise by computing actual lower bound functions. Again, we run (at query time) a backward search from t, this time computing for each vertex v a piecewise linear function $\pi_f\colon [0, M] \to \mathbb{R}_{\geq 0}$ mapping SoC to a lower bound on the trip time from v to t. Note that by construction, these functions are convex (and decreasing) for *single paths*. To simplify the search, we ignore battery constraints (thereby obtaining lower bounds on consumption). Scanning an arc a then corresponds to shifting the function by $-c(a)$ and $d(a)$ on the x-axis and y-axis, respectively. Moreover, rather than keeping sets of labels, we explicitly merge functions after scanning an arc (similar to time-dependent approaches [2]). Since merging may create functions that are no longer convex, we compute, after each merge operation, the *convex lower hull* of the result. While (slightly) deteriorating the quality of the bound, this reduces the number of breakpoints, and simplifies settling of charging stations (linking of two convex functions can be done in linear time). Again, it is easy to see that the resulting potential function $\pi_f\colon V \times [0, M] \to \mathbb{R}_{\geq 0}$, using the computed bounds, is consistent. The consistent key of a label ℓ (at vertex v) is $k(\ell) = \min\{\tau + \pi_f(v, b_\ell(\tau)) \mid \tau \geq 0\}$. Since both b_ℓ and $\pi_f(v)$ are piecewise linear, this minimum can be computed in a linear sweep over the breakpoints.

Note that computing the potential for every vertex is wasteful for short-range queries. Therefore, we suspend the backward search once it settled s. If we encounter a vertex v during the forward search that has not been covered by the backward search, we resume the backward search until a potential for v is determined.

Contraction Hierarchies. When adapting CH [16] to our scenario, vertex contraction becomes more expensive (each shortcut represents a pair of driving times and consumption profile, and we need shortcuts for all nondominated paths). Hence, we contract only some vertices (the *compo-*

Table 1: Preprocessing and query performance for different instances and battery capacity. We report preprocessing times in [m:s] for CHArge (which also applies to H_f and H_ω) and H_ω^A, the percentage of feasible paths, as well as heuristic and exact query times in [ms]. Results from [30] are as-is from a Core i3-2310M, 2.1 GHz.

			Preprocessing		Exact Query			Heuristic Query		
	Instance	M	CHArge	H_ω^A	Feasible	CHArge	[30]	H_f	H_ω	H_ω^A
Only BSS	Osg-r1000	100 km	11:37	2:30	100 %	122	539	94	38	3
Only BSS	Osg-r100	150 km	11:10	2:15	99 %	206	1 150	117	27	1
Only BSS	Osg-c643	100 km	11:21	2:32	98 %	326	—	283	48	3
Only BSS	Osg-c643	150 km	11:28	2:29	99 %	308	—	270	23	3
Only BSS	Ger-c1966	16 kWh	5:03	4:33	100 %	1 398	—	995	436	21
Only BSS	Ger-c1966	85 kWh	4:59	5:31	100 %	1 013	—	975	48	28
Only BSS	Eur-c13810	16 kWh	30:32	28:38	63 %	10 786	—	7 567	9943	207
Only BSS	Eur-c13810	85 kWh	30:16	29:47	100 %	47 921	—	35 060	1022	41
Mixed CS	Ger-c1966	16 kWh	5:03	4:33	100 %	8 629	—	1 495	1 357	155
Mixed CS	Ger-c1966	85 kWh	4:59	5:31	100 %	2 614	—	1 894	342	34
Mixed CS	Eur-c13810	16 kWh	30:32	28:38	63 %	24 148	—	10 039	17 630	2 694
Mixed CS	Eur-c13810	85 kWh	30:16	29:47	100 %	86 193	—	48 243	26 867	600

nent), leaving an uncontracted *core*. In particular, we keep all charging stations uncontracted [30]. Thus, complexity induced by charging stations only is contained within the core (simplifying the search in the component).

In order to reduce preprocessing time, we simplify our witness searches (possibly inserting unnecessary shortcuts, though without affecting correctness). We only search for single witnesses that dominate a shortcut candidate (hence, we might insert a shortcut that is dominated by a set of paths). During witness search we limit the number of labels per vertex to a small constant (10 in our experiments), deleting one of the two closest (wrt. driving time) whenever this size is exceeded. Finally, we prune the search after a fixed hop limit (20 in our experiments).

Since we compute a partial CH, the query algorithm consists of two phases. During the first, we run a (unidirectional) backward CH search from t, restricted to the component (pruning the query at core vertices). As the component contains no charging stations, a standard bicriteria search suffices. Note, however, that the SoC at t is yet unknown, hence, we compute consumption profiles instead of SoC values. The second phase runs a multi-target CFP query restricted to upward and core arcs.

CHArge. Combining CH and A* (restricting A* to the core), we obtain our fastest exact algorithm, CHArge. The query consists of three phases: a unidirectional (backward) phase from t in the component, a backward query in the (much smaller) core to compute potentials (π_ω or π_f), and a forward phase running (multi-target) CFP from s, restricted to upward and core arcs. As before, we suspend the second phase if the potential is known for all vertices in the (forward) queue. To decrease running time of the second phase, we preprocess lower bounds on core shortcuts for π_f.

6. HEURISTIC APPROACHES

With an $\mathcal{NP}$-hard problem at hand, we consider heuristic approaches based on CHArge, dropping optimality to reduce query times. When scanning multiple shortcuts (u, v) between u and v, we add at most one label to $L(v)$. This saves time for dominance checks and label insertions at $L(v)$, and reduces the number of labels in the queue. The idea is to use the potential at v to determine a shortcut minimizing the key of the new label, and add only this label to $L(v)$.

Our first heuristic, H_f, uses π_f to determine the best shortcut. For each shortcut (u, v), this requires a linear sweep over the breakpoints of $b_{\ell'}$ (ℓ' being the label created after scanning the shortcut) and $\pi_f(v)$ to find the best key. The idea of our second heuristic, H_ω, is to use π_ω, instead. However, if we ignore battery constraints and assume that we are not close to the target (i. e., $\beta_v < \text{dist}_c(v, t)$), the best shortcut (u, v) does not depend on the SoC at u, but only on $\text{dist}_\omega(u, t)$. Hence, for each pair of neighbors u and v we can precompute the optimal shortcut (the one that minimizes $\omega((u, v))$). During the query, we always use the precomputed shortcut for each neighbor v of u (instead of scanning all shortcuts). A third, even more aggressive variant, H_ω^A, uses the same idea during vertex contraction for CH, keeping only the ω-optimal shortcut for each pair of vertices. This significantly reduces the total number of shortcuts, allowing contraction of further vertices. The query of H_ω^A is similar to H_ω, however, since operating on a sparser graph, solutions may differ. Despite their heuristic nature, it is actually possible to prove that under certain circumstances, the heuristics H_ω and H_ω^A use an optimal shortcut. The basic idea is that, if charging is inevitable and charging at a rate of $\text{cf}_{\max}$ is possible when needed, then $\text{dist}_\omega(\cdot,\cdot)$ yields a tight bound on the trip time.

We also tried to combine CHArge with other well-known heuristics to reduce query complexity (e. g., ε-dominance and limiting label set sizes [4]). However, preliminary experiments showed that this drastically reduces solution quality.

Table 2: Instances. We report number of vertices, arcs, negative consumption arcs (as a fraction of total arcs), as well as charging stations (CS) obtained from ChargeMap. Note that the Osg instance has many degree-2 vertices, meant for visualization.

Ins.	# Vertices	# Arcs	# Arcs with $c < 0$	# CS
Ger	4 692 091	10 805 429	1 119 710 (10.36%)	1 966
Eur	22 198 628	51 088 095	6 060 648 (11.86%)	13 810
Osg	5 588 146	11 711 088	1 142 391 (9.75%)	643

Table 3: Detailed performance on Ger-c1966 85 kWh for Mixed and Realistic CS. For each query algorithm, we report settled labels, pairwise domination checks (Dom.) and runtime. For the resulting trips, we report the percentage of feasible and optimal paths, and average/maximum increase in trip time.

		Query			Trip Quality			
CS	Algorithm	# Labels	# Dom.	Time [ms]	Feasible	Optimal	Avg. $\times\tau_t$	Max. $\times\tau_t$
Mixed	CHArge	482 712	36 527 376	2 614	100 %	100 %	1.0000	1.0000
	H_f	443 134	139 897	1 894	100 %	85 %	1.0010	1.0725
	H_ω	190 955	5 578 309	342	100 %	80 %	1.0004	1.0213
	H_ω^A	11 309	29 695	34	100 %	52 %	1.0200	1.2387
Realistic	CHArge	395 841	48 611 726	2 457	100 %	100 %	1.0000	1.0000
	H_f	359 150	117 083	1 542	100 %	82 %	1.0007	1.0323
	H_ω	169 618	3 680 130	246	100 %	70 %	1.0009	1.0481
	H_ω^A	12 330	26 435	34	100 %	61 %	1.0128	1.1299

7. EXPERIMENTS

All implementations are 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, 128 GiB of DDR4-2133 RAM, 10 MiB of L3 and 256 KiB of L2 cache.

Instances. Our main instances are based on a road network of Europe (Eur) and the subnetwork of Germany (Ger), kindly provided by PTV AG (`ptvgroup.com`). As in [5], we extracted average speeds and categories of road segments, augmented by elevation data from the Shuttle Radar Topography Mission, v4.1 (`srtm.csi.cgiar.org`). We derived realistic energy consumption from a detailed micro-scale emission model [23], calibrated to a real Peugeot iOn. It has a battery capacity of 16 kWh, but we also evaluate for 85 kWh, as in high-end Tesla models. Moreover, we located charging stations on ChargeMap (`chargemap.com`). We also evaluate the largest instance of [30]: it uses OpenStreetMap data of Southern Germany (Osg) with SRTM, a basic physical consumption model, and 100–1,000 randomly chosen charging stations, but we also test on ChargeMap stations. See Table 2 for instance sizes. We consider several types of stations: battery swapping stations (BSS), superchargers (charging 50 % SoC in 20 min, 80 % in 40 min), and stations with fast (44 kW), medium (22 kW) and slow (11 kW) charging power (using the physical model of [32]), each approximated with 6 breakpoints (at $0, 80, 85, 90, 95, 100$ % SoC). We set arrangement time to 3 min for BSS, 1 min for all others.

Table 4: Impact of different core sizes on performance (Ger-c1966, 16 kWh).

Core size		Prepr.	Query [ms]	
Ø deg.	# Vertices	[h:m:s]	Only BSS	Mixed CS
8	344 066 (7.33%)	2:58	1 474.1	47 979.9
16	11 6917 (2.49%)	4:01	536.5	1 669.0
32	65 375 (1.39%)	5:03	436.1	1 356.8
64	43 036 (0.91%)	7:07	449.8	1 408.8
128	30 526 (0.65%)	11:16	509.6	1 585.4
256	22 592 (0.48%)	20:22	647.5	2 098.5
512	17 431 (0.37%)	37:11	880.7	2 739.9
1024	13 942 (0.29%)	1:05:51	1 264.6	3 934.2
2048	11 542 (0.24%)	2:00:27	1 822.6	5 670.1
4096	9 842 (0.20%)	4:17:36	2 706.6	8 420.1

Evaluation. We discuss preprocessing and query performance of our algorithms. We only report the fastest exact method (CHArge with π_f), for results on plain CFP see below. For each algorithm, we ran 1 000 queries between random source and target vertices, at maximum initial battery capacity, i. e., $\beta_s = M$. Table 1 shows detailed figures, organized in three blocks: the first considers instances from [30], using only BSS, randomly placed charging stations (r100, r1000) and battery capacities that translate to a certain range in the physical model. The second block also considers only BSS, but ChargeMap stations and the Peugeot iOn model for different battery capacities. The third block additionally uses a mixed composition of chargers (10 % BSS, 20 % super chargers, 30 % fast chargers, 40 % slow chargers).

Preprocessing times are quite practical, considering the problem at hand, ranging from about 5–30 minutes. Table 4 shows details on preprocessing effort and query performance for different core sizes on the Germany instance (16 kWh). Vertex contraction was stopped as soon as the average vertex degree in the core reached the threshold shown in the first column. We report resulting core sizes and preprocessing time, as well as query times of H_ω (the fastest query variant that uses the same core as CHArge) for the BSS and mixed station composition, respectively. While preprocessing effort increases with decreasing core size, we achieve best query performance at average core degree 32, hence we always stop vertex contraction at this threshold. The corresponding relative core sizes are 1.3–1.7 % on PTV and 0.3–0.4 % on OpenStreetMap data.

Regarding query performance (see Table 1), we observe that our approach is considerably faster than that of [30], even when accounting for differences in hardware. At the same time, CHArge is more general and not inherently restricted to BSS. Furthermore, a denser distribution of random charging stations enables faster query times (since the proposed potentials more often rightly assume that a station will be available along the path). We see that using ChargeMap station locations (Osg-c643) gives a slightly harder instance. Overall, we are able to solve instances that are harder than those evaluated in [30].

For the instances representing larger networks (Ger and Eur), the mixed scenarios are (slightly) harder to solve (as vertex potentials are less accurate). On the other hand, increasing the maximum battery capacity leads to faster queries: less charging is required, hence goal direction is more helpful. A notable exception is the 16 kWh battery

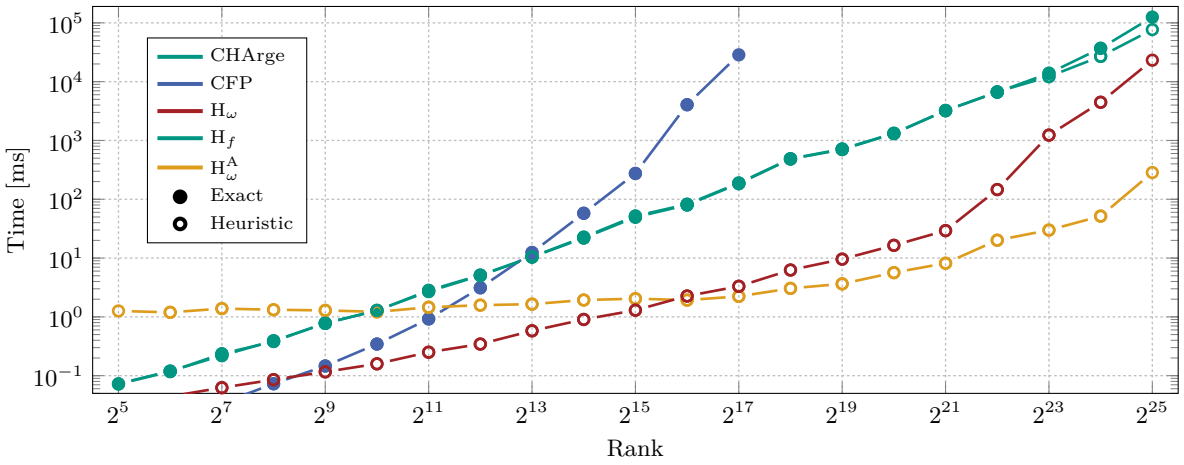

Figure 4: Scalability of our approaches on Europe with a 85 kWh battery and Mixed CS.

on the Europe instance. In this instance, the number of feasible queries drops significantly, due to a highly non-uniform distribution of charging stations (sparse in parts of Southern and Eastern Europe). This benefits approaches based on π_f-potentials (as we can detect infeasibility already during potential computation), but often deteriorates the performance of π_ω-based algorithms (because the target is never reached, hence large parts of the graph are explored until the queue is empty). All in all, running times of the exact algorithm are below 10 s on average on all Germany instances and below 90 s on Europe, which is quite notable given that we could not even run a single (long-distance) CFP query on this instance in several hours. Also, the Mixed CS configuration was the hardest configuration in our evaluation (we also tried other configurations, e. g., without BSS). For heuristic approaches, we see that (in contrast to CHArge), those based on π_ω are faster (except for instances with many infeasible queries).

Table 3 reports detailed query times of the Germany instance with 85 kWh battery, which contains the most sensible queries: due to a reasonably dense distribution of charging stations, all queries are feasible, and the target is always reachable with a small number of charging stops (in contrast to the harder queries across Europe, which we chose rather to analyze scalability of our algorithms). We also add numbers for a (currently) more realistic scenario, containing no BSS, 20 % superchargers, 40 % fast and medium regular stations, respectively. We see that the number of queue extractions is a good indicator of the running time of all approaches. Moreover, all approaches are quite practical: computing optimal routes takes only a few seconds. The heuristic H_ω provides a good trade-off between running times (some 300 ms) with small errors (below 0.1 % on average, below 5 % in the worst case). Our aggressive approach H_ω^A allows query times of 34 ms on average, which is fast enough even for real-time applications, however, we see that solution quality is up to 24 % off the optimum in the worst case. Still, the average error of all (CHArge-based) heuristics is very low, and the optimal solution is found in more than half of the cases.

Figure 4 shows median times on our hardest instance, distributed by their Dijkstra rank (the logarithm of the number of queue extractions when running Dijkstra from s to t on unconstrained driving distance [2]). We ran 100 queries per rank. Query times for CFP are only reported up to a rank of 17, for higher ranks at least one query did not terminate within an hour. Moreover, charging becomes necessary at a Dijkstra rank of (roughly) 22 — running times increase significantly at this rank for our faster approaches. It is also clear that all approaches have exponential performance (while still being practical even for ranks beyond 20).

Turn Costs. Finally, we also considered graphs enriched with turn costs, i. e., additional time and energy consumption to take account of acceleration and deceleration at turns (or when speed limits change). Using a simple model to assign time and consumption overhead to turns, we modified our input instances according to the edge-based model of [17]. For the enriched network of Europe, CH preprocessing took slightly longer (37 min, 35 s), resulting in a smaller relative core size (1.2 %). Query times were also similar to the standard case, e. g., below 89 s on average on the hardest instance (Europe, 85 kWh). This clearly indicates that our approach can be readily extended to handle turn costs and turn restrictions.

8. CONCLUSION

We introduced CFP, a new approach to route planning for EVs that computes shortest feasible paths with charging stops, minimizing overall trip time. Combining vertex contraction and charging-aware goal-directed search, we introduced CHArge, which solves this $\mathcal{NP}$-hard problem optimal and with practical performance even on large realistic inputs.

Future work includes allowing multiple driving speeds (and consumption values) per road segment, e. g., in order to save energy on the highway by driving at reasonable lower speeds [4, 19, 22]. Moreover, we are interested in more detailed models for energy consumption on turns, as well as historic and live traffic. To allow for the latter, the adaptation of customization techniques appears to be a natural extension of our approach. While preliminary experiments showed that CHArge performs equally fast on vertex orders required by Customizable Contraction Hierarchies [9], a major challange is the design of fast customization algorithms that can handle the dynamic data structures required by label sets in our approach. On the other hand, preprocessing time for CHArge is already in the order of minutes (and can

even be reduced by increasing core size at the cost of query times, see Table 4). This is fast enough for many applications that require frequent updates of the underlying graph.

From a theoretical perspective, approximability is an open question. Recent results [27] show that an FPTAS exists for a very similar problem, which might extend to our setting.

Acknowledgements. We thank Raphael Luz for providing consumption data, Martin Uhrig for charging functions, and Sabine Storandt for benchmark data.

9. REFERENCES

[1] I. Abraham, D. Delling, A. Fiat, A. V. Goldberg, and R. F. Werneck. HLDB: Location-Based Services in Databases. In: *ACM SIGSPATIAL'12*, pp. 339–348. ACM, 2012.

[2] H. Bast, D. Delling, A. V. Goldberg, M. Müller–Hannemann, T. Pajor, P. Sanders, D. Wagner, and R. F. Werneck. Route Planning in Transportation Networks. Technical Report abs/1504.05140, ArXiv e-prints, 2015.

[3] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining Hierarchical and Goal-directed Speed-up Techniques for Dijkstra's Algorithm. *ACM JEA*, 15(2.3):1–31, 2010.

[4] M. Baum, J. Dibbelt, L. Hübschle-Schneider, T. Pajor, and D. Wagner. Speed-Consumption Tradeoff for Electric Vehicle Route Planning. In: *ATMOS'14*, pp. 138–151. OASIcs, 2014.

[5] M. Baum, J. Dibbelt, T. Pajor, and D. Wagner. Energy-Optimal Routes for Electric Vehicles. In: *ACM SIGSPATIAL'13*, pp. 54–63. ACM, 2013.

[6] D. Delling, A. V. Goldberg, T. Pajor, and R. F. Werneck. Customizable Route Planning in Road Networks. *Transp. Sci.*, 2015.

[7] D. Delling, A. V. Goldberg, and R. F. Werneck. Faster Batched Shortest Paths in Road Networks. In: *ATMOS'11*, pp. 52–63. OASIcs, 2011.

[8] D. Delling and R. F. Werneck. Customizable Point-of-Interest Queries in Road Networks. In: *ACM SIGSPATIAL'13*, pp. 490–493. ACM, 2013.

[9] J. Dibbelt, B. Strasser, and D. Wagner. Customizable Contraction Hierarchies. In: *SEA'14*, LNCS 8504, pp. 271–282. Springer, 2014.

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

[11] A. Efentakis and D. Pfoser. Optimizing Landmark-Based Routing and Preprocessing. In: *IWCTS'13*, pp. 25:25–25:30. ACM, 2013.

[12] J. Eisner, S. Funke, and S. Storandt. Optimal Route Planning for Electric Vehicles in Large Networks. In: *AAAI*, pp. 1108–1113. AAAI, 2011.

[13] S. Funke and S. Storandt. Polynomial-Time Construction of Contraction Hierarchies for Multi-Criteria Objectives. In: *ALENEX'13*, pp. 31–54. SIAM, 2013.

[14] R. Geisberger, M. Kobitzsch, and P. Sanders. Route Planning with Flexible Objective Functions. In: *ALENEX'10*, pp. 124–137. SIAM, 2010.

[15] R. Geisberger, M. Rice, P. Sanders, and V. Tsotras. Route Planning with Flexible Edge Restrictions. *ACM JEA*, 17(1):1–20, 2012.

[16] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transp. Sci.*, 46(3):388–404, 2012.

[17] R. Geisberger and C. Vetter. Efficient Routing in Road Networks with Turn Costs. In: *SEA'11*, LNCS 6630, pp. 100–111. Springer, 2011.

[18] A. V. Goldberg and C. Harrelson. Computing the Shortest Path: A* Search Meets Graph Theory. In: *SODA'05*, pp. 156–165. SIAM, 2005.

[19] M. T. Goodrich and P. Pszona. Two-Phase Bicriterion Search for Finding Fast and Efficient Electric Vehicle Routes. In: *ACM SIGSPATIAL'14*, pp. 193–202. ACM, 2014.

[20] G. Y. Handler and I. Zang. A Dual Algorithm for the Constrained Shortest Path Problem. *Networks*, 10(4):293–309, 1980.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

[22] F. Hartmann and S. Funke. Energy-Efficient Routing: Taking Speed into Account. In: *KI'14*, LNCS 8736, pp. 86–97. Springer, 2014.

[23] S. Hausberger, M. Rexeis, M. Zallinger, and R. Luz. Emission Factors from the Model Phem for the HBEFA Version 3. TR I-20/2009, Univ. of Technology, Graz, 2009.

[24] D. B. Johnson. Efficient Algorithms for Shortest Paths in Sparse Networks. In: JACM, 24(1):1–13, 1977.

[25] C. Liu, J. Wu, and C. Long. Joint Charging and Routing Optimization for Electric Vehicle Navigation Systems. In: IFAC'14, pp. 2106–2111. IFAC, 2014.

[26] E. Q. V. Martins. On a Multicriteria Shortest Path Problem. *Eur. J. Oper. Res.*, 16(2):236–245, 1984.

[27] S. Merting, C. Schwan, and M. Strehler. Routing of Electric Vehicles: Constrained Shortest Path Problems with Resource Recovering Nodes. In: *ATMOS'15*, pp. 29–41. OASIcs, 2015.

[28] M. Sachenbacher, M. Leucker, A. Artmeier, and J. Haselmayr. Efficient Energy-Optimal Routing for Electric Vehicles. In: *AAAI*, pp. 1402–1407. AAAI, 2011.

[29] O. J. Smith, N. Boland, and H. Waterer. Solving Shortest Path Problems with a Weight Constraint and Replenishment Arcs. *Comput. Oper. Res.*, 39(5):964–984, 2012.

[30] S. Storandt. Quick and Energy-Efficient Routes: Computing Constrained Shortest Paths for Electric Vehicles. In: *ACM SIGSPATIAL'12*, pp. 20–25. ACM, 2012.

[31] T. M. Sweda, I. S. Dolinskaya, and D. Klabjan. Adaptive Routing and Recharging Policies for Electric Vehicles. Northwestern University, Illinois, Working Paper No. 14-02, 2014.

[32] M. Uhrig, L. Weiß, M. Suriyah, and T. Leibfried. E-Mobility in Car Parks – Guidelines for Charging Infrastructure Expansion Planning and Operation Based on Stochastic Simulations. In: *EVS28*, 2015.