Robustness Generalizations of the Shortest Feasible Path Problem for Electric Vehicles

Payas Rajan
Department of Computer Science & Engineering, University of California, Riverside, CA, USA

Moritz Baum
Apple, Cupertino, CA, USA

Michael Wegner
Apple, Cupertino, CA, USA

Tobias Zündorf
Apple, Cupertino, CA, USA

Christian J. West
Apple, Cupertino, CA, USA

Dennis Schieferdecker
Apple, Cupertino, CA, USA

Daniel Delling
Apple, Cupertino, CA, USA

Abstract

Electric Vehicle routing is often modeled as a Shortest Feasible Path Problem (SFPP), which minimizes total travel time while maintaining a non-zero State of Charge (SoC) along the route. However, the problem assumes perfect information about energy consumption and charging stations, which are difficult to even estimate in practice. Further, drivers might have varying risk tolerances for different trips. To overcome these limitations, we propose two generalizations to the SFPP; they compute the shortest feasible path for *any* initial SoC and, respectively, for *every* possible minimum SoC threshold. We present algorithmic solutions for each problem, and provide two constructs: *Starting Charge Maps* and *Buffer Maps*, which represent the tradeoffs between robustness of feasible routes and their travel times. The two constructs are useful in many ways, including presenting alternate routes or providing charging prompts to users. We evaluate the performance of our algorithms on realistic input instances.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Mathematics of computing → Paths and connectivity problems

Keywords and phrases Electric Vehicles, Route Planning

Digital Object Identifier 10.4230/OASIcs.ATMOS.2021.11

Funding *Payas Rajan*: This work was done while the author was an intern at Apple Inc.

1 Introduction

Several factors can cause an Electric Vehicle (EV) to get stranded along a route: They often have shorter ranges than internal combustion (IC) vehicles, charging stations can be sparse and fragmented among different providers. For drivers, this *stranding risk* manifests as *range anxiety* and *range stress* [18, 25, 40, 41, 44, 46]. To alleviate range anxiety, route planning for EVs must consider battery constraints while selecting routes [19, 33, 34, 48, 49].

Previous work [7, 8] models EV routing with charging stops as the NP-hard Shortest Feasible Path Problem (SFPP): Given a road network modeled as a weighted, directed graph with energy consumptions and travel times on each edge; charging stations on a subset of vertices and their respective concave charging functions; a source vertex, a destination vertex and a starting battery SoC, find a path that minimizes the total travel time including charging time while maintaining a non-zero battery SoC at all points along the route. The Charging Function Propagation (CFP) algorithm solves SFPP in exponential time and space.

In practice, however, the shortest feasible path might not be sufficient. First, the energy consumptions on edges are derived from estimation models that are not perfectly accurate [14, 20, 42, 43]. Second, the energy consumption of an EV depends on several factors that

are difficult to even estimate: driver aggressiveness, age of the battery, wear and tear of the EV. Each of these factors can affect the energy consumption significantly [1, 22]. Third, users may have varying risk tolerances, and thus a one-size-fits-all approach is not sufficient to alleviate range anxiety when serving routes for a large number of EV drivers.

In this work, we introduce two generalizations of SFPP which are used to compute two constructs, the *Starting Charge Map (SCM)* and *Buffer Map (BM)*. Both SCM and BM are computed between a source vertex s and target vertex t. Evaluating SCM_{st} for a valid starting SoC β_s gives the corresponding shortest feasible path between s and t, while evaluating BM_{st} for buffer energy $\mathfrak{b}$ returns a shortest feasible path where the SoC is guaranteed to never drop below $\mathfrak{b}$ along the route.

The SCM and BM allow route planning systems to access a larger set of alternative feasible paths than the standard CFP algorithm, which returns only a single feasible path. This variety in paths has several applications–recommending EV drivers alternative routes, generating suggestions like charging extra at s to save travel time, or letting users choose the degree of acceptable risk for a trip. Both problems can be solved by brute force approaches that run CFP for all possible values of β_s or $\mathfrak{b}$. However, since β_s or $\mathfrak{b}$ can take an infinite number of possible values, such an approach would simply not terminate. In this paper, we make the following contributions:

- We introduce the *Starting Charge Map (SCM)* and *Buffer Map (BM)* that encapsulate a set of alternative routes to help alleviate range anxiety for a wide variety of EV drivers.
- Computing SCM and BM using standard CFP requires several expensive runs of the algorithm. We present fast, exact algorithms that compute the two abstractions with acceptable real-time performance on large graphs.
- We evaluate our algorithms on realistic instances, using real-world road networks of California and Oregon, an energy consumption model taken from a Nissan Leaf 2013 [20], and a dataset of public EV charging stations [2]. Our results show good performance even without the use of preprocessing techniques for shortest path queries.

2 Related Work

Most current EVs suffice for a majority of trips that drivers take, as shown in [39]. However, *range anxiety*, defined as an EV driver's fear of getting stranded along a route is often cited as a major hindrance to widespread EV adoption [24, 26]. Prior work shows that perceived range anxiety is inversely related to the degree of drivers' *trust* in the EVs [31, 44, 25, 46, 32]. Route planning for EVs, therefore, has two objectives: Minimize travel times under battery constraints, and reduce *surprise* for the driver to minimize range anxiety.

Early works on EV route planning like [3, 45] consider the problem of minimizing energy consumption along routes insted of standard route planning formulations that minimize travel time [4]. Since then, many additions have been proposed to the EV routing problem to make it more realistic. Several newer variants consider battery-swapping stations [19] or charging functions [7, 12, 37, 50]. Some works [29, 48, 49] model EV routing as a multicriteria Dijkstra's search [35], which returns a set of pareto-optimal routes that are not dominated in either travel time or energy consumption. Conversely, some other works like [7, 12] present EV routing as an extension of the Constrained Shortest Path problem. These problems seek to minimize total travel time including charging time, while constraining the total energy consumption of paths to levels allowed by realistic battery capacities. Another line of research considers "profile queries", which look for all optimal shortest paths depending on a certain state [47], e.g., the initial state of charge of an EV [10, 13] or the current point in time [11, 17, 23].

Underlying all EV routing algorithms is an assumption that the energy consumptions assigned to graph edges are accurate. In practice, this is difficult to achieve with existing energy consumption models [14, 20, 21, 42, 43, 36]. EV energy consumption is affected by several factors including traffic conditions, driver aggressiveness, battery health and regular wear-and-tear of the vehicle, which are hard to estimate. Recently, [1] showed that each of these factors can impact the energy consumption along short routes by as much as 40%. Similarly, [43] show a high variance in EV energy consumptions for short trips. To mitigate the effects of inaccurate estimates, [46] recommend holding a *safety margin* between 12 and 23% of battery capacity. Only few EV routing algorithms [22, 30] accommodate buffer energy for variance in energy consumption estimates or provide robust routes.

3 Preliminaries

Our setup is similar to the standard shortest feasible path problem [7, 8]. We consider a road network modeled as directed graph $G = \langle V, E \rangle$, with V the set of vertices and $E : V \times V$ the set of edges. We are given two edge weight functions $d : E \to \mathbb{R}_{\geq 0}$ and $c : E \to \mathbb{R}$ that assign the travel time and energy consumption to each $e \in E$. An $s - t$ *path* in G is a sequence of adjacent vertices $P = [s = v_1 v_2 \dots v_n = t]$, such that $\forall 1 \leq i \leq n, (v_i, v_{i+1}) \in E$ holds.

For a path P, the total *driving time* is $d(P) = \Sigma_{i=1}^{n-1} d(v_i, v_{i+1})$. The *consumption profile*, $f_P : [0, M] \to [-M, M] \cup \{-\infty\}$ is a function that maps the starting SoC β_s to residual SoC β_t at t after traversing P. f_P can be negative due to energy recuperation along P, or $-\infty$ if it is not possible to traverse P with starting SoC β_s. $f_P(\beta)$ can be computed using a 3-tuple $\langle \text{in}_P, \text{cost}_P, \text{max}_P \rangle$, where in_P is the minimum SoC required at s to traverse P, $cost_P = \Sigma_{i=1}^{n-1} c(v_i, v_{i+1})$, and out_P is the maximum SoC possible at t after traversing P [19]. Conversely, we define an *inverse consumption profile* $f_P^{-1} : [-M, M] \to [0, M] \cup \{\infty\}$, which maps residual SoC β_t to the starting SoC β_s. We evaluate both functions as:

$$f_P(\beta) = \begin{cases} -\infty & \text{if } \beta < \text{in}_P \\ \text{out}_P & \text{if } \beta - \text{cost}_P > \text{out}_P \\ \beta - \text{cost}_P & \text{otherwise} \end{cases}, \quad f_P^{-1}(\beta) = \begin{cases} \infty & \text{if } \beta > \text{out}_P \\ \text{in}_P & \text{if } \beta + \text{cost}_P < \text{in}_P \\ \beta + \text{cost}_P & \text{otherwise} \end{cases}$$

Let $f_\phi(\beta)$ and $f_\phi^{-1}(\beta)$ be identity SoC profiles that always map a given SoC β to itself. Given two paths $P = [v_1 v_2 \dots v_k]$ and $Q = [v_{k+1} \dots v_n]$, we can get the concatenation $P \circ Q = [v_1 \dots v_k v_{k+1} \dots v_n]$ and a linked consumption profile $f_{P \circ Q}$ as $\text{in}_{P \circ Q} = max\{\text{in}_P, \text{cost}_P + \text{in}_Q\}$, $\text{out}_{P \circ Q} = min\{\text{out}_Q, \text{out}_P - \text{cost}_Q\}$, and $\text{cost}_{P \circ Q} = max\{\text{cost}_P + \text{cost}_Q, \text{in}_P - \text{out}_Q\}$, if $\text{out}_P \geq \text{in}_Q$; otherwise, $P \circ Q$ is infeasible and $f_{P \circ Q} \equiv -\infty$. Lastly, an (inverse) SoC profile f_1 is said to *dominate* f_2 if $\forall \beta \in [0, M], f_1(\beta) \geq f_2(\beta)$.

A set $S \subseteq V$ marks the available charging stations on the road network. Each $v \in S$ is assigned a concave, monotonically increasing *charging function* $\text{cf}_v : \mathbb{R}_{\geq 0} \to [0, M]$ that maps the charging time at v to the resultant SoC after charging. Conversely, we also define the inverse charging function $\text{cf}_v^{-1} : [0, M] \to \mathbb{R}_{\geq 0}$. To obtain the time it takes to charge from β_1 to β_2, we compute $\text{cf}^{-1}(\beta_2) - \text{cf}^{-1}(\beta_1)$.

▶ **Definition 1.** *A* shortest feasible path P *between a source* $s \in V$ *and a target* $t \in V$ *for an EV with a starting SoC* $\beta_s \in [0, M]$ *is one that minimizes the total trip time (travel time + charging time) while maintaining a non-negative battery SoC at all points on* P*.*

For this work, we add two constraints to the original definition of charging functions: First, we require that all charging functions have a minimum initial SoC of 0 and are able to fully charge EVs to M SoC. This constraint is realistic as any real-world charging station can charge an EV with an empty battery to its full capacity. Second, similar to [7], we require all charging functions to be *piecewise linear*.

3.1 Charging Function Propagation (CFP)

CFP [7, 8] is a generalization of the bicriteria Dijkstra's algorithm [35] with two major differences: First, the set of labels at a vertex represent all possible tradeoffs between charging time and the resultant SoC after charging at the last station, and second, the decision how much to charge at a station is taken at the immediately following station the EV visits. This is because the amount of charge needed at $u \in S$ is dependent on energy consumed by the EV between u and the next station $v \in S$. If v_i and v_j are two consecutive charging stations on a path $P = [v_1 \dots v_n]$, we call the subpath $[v_i \dots v_j]$ a *leg* of P.

Assume that we want to find a shortest feasible path between $s, t \in V$ for starting SoC β_s. For all $v \in V$, we maintain sets $L_{uns}(v)$ for unsettled and $L_{set}(v)$ for settled labels. For vertex v, a label of the CFP search is a 4-tuple $\ell = \langle \tau_v, \beta_u, u, f_{[u \dots v]} \rangle$ where τ_v is the total travel time from s to v except the charging time at the last charging station u, β_u is the EV's SoC on arriving at u and $f_{[u \dots v]}$ is the consumption profile of subpath $[u \dots v]$. The CFP search propagates through G as follows:

1. *At s*: A label $\ell = \langle 0, \beta_s, s, f_\phi \rangle$ is added to the travel time ordered min-priority queue PQ.
2. *Search reaches a non-charging vertex $v \neq t$*: Let path $P = [s = v_1 \dots v_k = v]$ and total travel time $\tau_P = \Sigma_{i=1}^{k-1} d(v_i, v_{i+1})$. Create label $\langle \tau_P, \beta_s, s, f_P \rangle$ and add to $L_{uns}(v)$.
3. *Search reaches first charging station vertex $v \neq t$*: Let path $P = [s \dots v]$ and total travel time over P be τ_P. Create label $\langle \tau_P, f_P(\beta_s), v, f_\phi \rangle$ and add to $L_{uns}(v)$.
4. *Search reaches a non-charging vertex $v \neq t$*: Let $\ell = \langle \tau_v, \beta_u, u, f_{[u \dots v]} \rangle$ be the current label extracted from PQ. Since u is the last charging station, let subpath $P = [u \dots v]$ and the total travel time over P be τ_P. Add label $\langle \tau_{[s \dots v]}, f_{[s \dots v]}(\beta_s), u, f_P \rangle$ to $L_{uns}(v)$.
5. *Search reaches a subsequent charging vertex $v \neq t$*: Let $\ell = \langle \tau_v, \beta_u, u, f_{[u \dots v]} \rangle$ be the current label extracted from PQ. Since u is the last charging station, let *leg* $\mathcal{L} = [u \dots v]$ of path $P = [s \dots v]$, and the total travel time over P be τ_P. Compute the *SoC function* $b_\ell(\tau) := \tau_P + f_{\mathcal{L}}(\mathrm{cf}_u(\beta_u, \tau - \tau_P))$. Since all charging functions are assumed to be piecewise linear, it suffices to create one label per breakpoint of b_ℓ [7]. For breakpoint $B = (\tau_B, SoC_B)$, create a label $\langle \tau_B, SoC_B, v, f_\phi \rangle$ and add to $L_{uns}(v)$.
6. *Search reaches destination t*: Terminate and backtrack to extract a path from s to t.

The label sets for all $v \in V$ are used to minimise the total number of dominance checks among labels for v. L_{uns} is implemented as a min-heap with total feasible travel time as the key, and the following invariant is maintained: The minimum label ℓ in $L_{uns}(v)$ is not dominated by any label in $L_{set}(v)$. Label ℓ dominates ℓ' iff $b_\ell(\tau) \geq b_{\ell'}(\tau)$ when $\tau \geq 0$.

As the number of labels created during CFP search can be exponential, the algorithm belongs to the EXPTIME class. A combination of A* search using potential functions and Contraction Hierarchies [28] can be used to speed up CFP on large graphs in practice. When both speedup techniques are combined, the result is called the CHArge algorithm [7, 8].

4 Starting Charge Maps

▶ **Definition 2.** *For a given source $s \in V$ and target $t \in V$, a* starting charge map *$SCM_{st} : [0, M] \to P$ is a function that maps a starting charge β_s to the corresponding shortest feasible path P.*

An *SCM* is a generalization of the shortest feasible path problem where the starting SoC β_s is unknown. First, it can be used to *recommend* users faster routes that they can take if the starting SoC is higher. For example, given an *SCM*, it is trivial to generate recommendations for EV drivers like "The best path with your current SoC takes 45 minutes,

but you might save 10 minutes if you spend 15 more minutes charging at your present location before starting your trip". Such recommendations can be particularly useful to EV drivers for routes with flexible starting times. Second, different trips taken by an EV user might have *different levels of risk aversion*, and an SCM can be used to show users feasible paths that suit the current scenario. As an example, consider two EV trips, the first through an urban area with a high density of charging stations during daytime, and a second trip through a sparsely populated area after nightfall. In the first scenario, most drivers might trade off a higher stranding risk for shorter travel times, while the preferences might be reversed for the second route. SCMs can be used to explore such alternatives and present them to the driver. Asking the driver to charge longer might reduce the risk, while allowing to start with a lower SoC usually increases the risk. Lastly, in most applications, routes are computed on a server and sent to the users on mobile clients. Since battery constraints apply for EV routing, more information about the vehicle needs to be sent to the server than for regular Internal Combustion (IC) vehicles. If instead of individual routes, SCMs are computed and sent to the client for display, the current SoC no longer needs to be sent to the routing server, which may result in *better privacy* for the drivers.

A brute force approach to compute SCM_{st} is to run the CFP algorithm for all values in $[0, M]$. However, since $[0, M]$ contains an infinite number of values, this is clearly not feasible. Even if we discretize the domain and restrict it to only percentage values that are multiples of a small fixed integer k, running CFP $\frac{100}{k}$ times, once each for $\{0, k, 2k, 3k, ..., 100\}\%$, would still be too slow for interactive routing applications where queries need to be answered quickly. A better approach is to run a series of binary searches in the starting SoC range $[0, M]$ such that on iteration i, the search returns a breakpoint starting SoC $\beta \in [0, M]$, where the shortest feasible paths for starting SoC β and $(\beta + \epsilon)$ differ by at least one edge. However, if $|SCM_{st}| = N$, such an approach would take $N \log N$ runs of the CFP algorithm. In the next section, we present an algorithm that computes SCM_{st} in N runs.

4.1 Reverse Charging Function Propagation

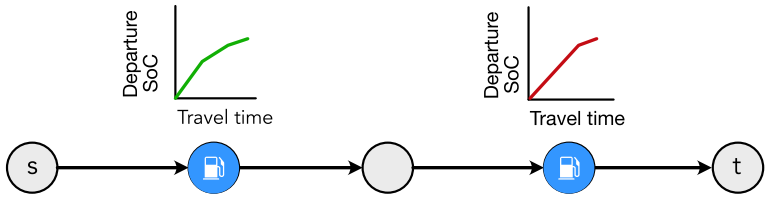
First, we introduce the following intermediate problem:

▶ **Definition 3.** *The Reverse Shortest Feasible Path (RSFP) Problem:*
Given a graph $G = \langle V, E \rangle$, edge weight functions $d : E \to \mathbb{R}_{\geq 0}$ and $c : E \to \mathbb{R}$ that represent travel time and energy consumption on edges respectively, a source $s \in V$ and a target $t \in V$, a set $S \subseteq V$ marked as charging stations, and an SoC β_t, find a shortest path P such that SoC never drops below 0 along P and has a residual SoC at least β_t at t.

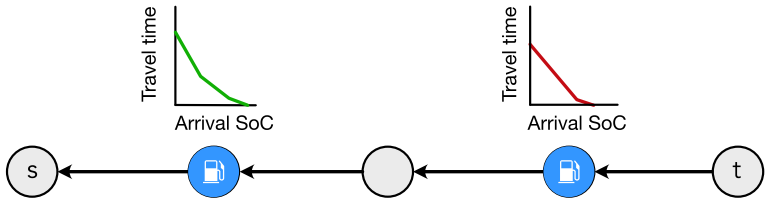
As RSFP is closely related to the regular shortest feasible path problem, it can be solved with a *reverse* variant of the CFP algorithm. Note that several operations needed for CFP are not symmetric, e.g., $f(P \circ Q) \neq f(Q \circ P)$. Following, we detail the Reverse Charging Function Propagation (RCFP) algorithm and extend it to compute Starting Charge Maps.

The Reverse CFP works on a backward graph G', obtained by reversing the directions of all edges in G. The RCFP search starts at t with residual SoC β_t and propagates towards s. At $v \in V$, a label ℓ' is defined as $\langle \tau_t, \beta'_u, u, f_{[v \dots u]} \rangle$, with τ_t the total travel time on subpath $[t \dots v]$, u the last charging station encountered in the search, β'_u the SoC *after* charging at u, and $f_{[v \dots u]}$ the consumption profile of subpath $[v \dots u]$.

A key difference between forward and reverse CFP search labels is that while a label ℓ for the forward search contains β_u, the SoC *before* charging at the last charging station u, ℓ' stores β'_u, the SoC *after* charging at u. Computing β'_u is *only* possible in reverse CFP search, because of the following: As forward CFP search reaches v, only the exact energy



(a) CFP runs on G with piecewise linear, concave charging functions.



(b) RCFP runs on backward graph G' with inverse charging functions.

Figure 1 Comparing SFP and RSFP problem setups. While charging functions map the time spent charging to an EV's SoC at *departure*, inverted charging functions map the EV's SoC at *arrival* at the charging station to the least possible charging time required to reach target.

consumption on $[u \dots v]$ is known, and therefore CFP needs to keep track of all possible charging scenarios at *previous* charging station u until the search reaches t or the next charging station. However, in RCFP search, the exact energy consumption between v and the target or *next* charging station u is known. Thus, as RCFP search reaches a charging station or origin, we know *exactly* how much charge is needed to travel from v to u, and have residual SoC β'_u. Both forward and reverse CFP maintain two label sets for each $v \in V$: $L_{uns}(v)$ for unsettled and $L_{set}(v)$ for settled labels.

Further, for RCFP, we transform all cf_v to inverse charging functions cf_v^{-1}. At $v \in S$, cf_v^{-1} returns the time required to charge an empty battery to resultant SoC β'. Note that under our assumptions, the inverse charging functions are piecewise linear, convex and monotonically decreasing. RCFP propagates through G' as follows:

1. *At t*: A label $\ell' = \langle 0, \beta_t, t, f_\phi^{-1} \rangle$ is added to the travel time ordered min-priority queue.
2. *Search reaches a non-charging vertex $v \neq s$*: Let path $P = [v = v_1 \dots v_k = t]$ and total travel time be $\tau_P = \Sigma_1^k d(v_i, v_{i+1})$. Create label $\langle \tau_P, \beta_t, t, f_P^{-1} \rangle$ and add to $L_{uns}(v)$.
3. *Search reaches first charging station vertex $v \neq s$*: Let path $P = [v \dots t]$, total travel time over P be τ_P. Create label $\langle \tau_P, f_P^{-1}(\beta_t), v, f_\phi^{-1} \rangle$ and add to $L_{uns}(v)$.
4. *Search reaches a non-charging vertex $v \neq t$*: Let $\ell = \langle \tau_v, \beta_u, u, f_{[u \dots v]} \rangle$ be the current label extracted from PQ. Since u is the last charging station, let subpath $P = [u \dots v]$ and the total travel time over P be τ_P. Add label $\langle \tau_{[s \dots v]}, f_{[s \dots v]}^{-1}(\beta_s), u, f_P^{-1} \rangle$ to $L_{uns}(v)$.
5. *Search reaches a subsequent charging vertex $v \neq s$*: Let $\ell' = \langle \tau_t, \beta'_u, u, f_{[u \dots v]}^{-1} \rangle$ be the current label extracted from PQ. Since u is the last charging station, let *leg* $\mathcal{L} = [u \dots v]$ and path $P = [v \dots t]$. Let the total travel time over P be τ_P. Next, compute the *Starting SoC function* $b'_{\ell'}(\beta) := \tau_P + max(0, \mathrm{cf}_u^{-1}(f_\ell^{-1}(\beta)) - \mathrm{cf}_u^{-1}(\beta'_u))$. Again, since we assume that all inverted charging functions are piecewise linear, it suffices to create one label per breakpoint of $b'_{\ell'}$. For breakpoint $B = (\tau_B, SoC_B)$, create a label $\langle b'_{\ell'}(SoC_B), SoC_B, v, f_P^{-1} \rangle$ and add to $L_{uns}(v)$.
6. *Search reaches destination s*: Terminate and backtrack to extract a path from t to s.

A label ℓ'_1 is said to dominate ℓ'_2 iff $b'_{\ell'_1}(\beta) \leq b'_{\ell'_1}(\beta)$ for $\beta \geq 0$.

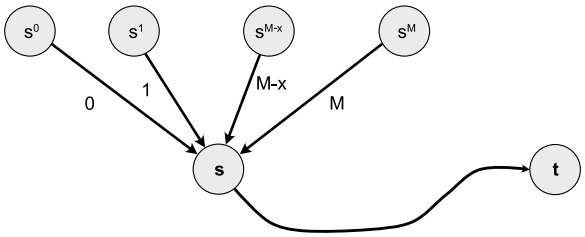


Figure 2 "Virtual" vertices added to the graph.

▶ **Lemma 4.** *If a shortest feasible $s-t$ path exists, running the RCFP algorithm from t to s with $\beta_t = 0$ finds it.*

Proof. Let P be a shortest feasible $s-t$ path in G. Now, we show that RCFP computes the correct solution (travel time and starting SoC) for P. We distinguish three cases:

- *P contains no charging stop:* The linking operation on (inverse) consumption profiles is associative [10]. Further, the order in which labels are added to $L_{uns}(v)$ does not affect the correctness of the algorithms. Therefore, a shortest feasible path is found regardless of search direction and the correctness of RCFP follows from that of CFP [8].
- *P contains a single charging stop:* Let u be the charging stop on P, which divides P into subpaths $[s \dots u]$ and $[u \dots t]$. As the RCFP search starts from t and reaches u, the departure SoC at u is set to $\text{in}_{[u \dots t]}$, the minimum SoC required to ensure feasibility of P. Charging more at u only increases the charging time without any corresponding decrease in travel time, which in turn increases the total travel time along P, violating the assumption that P is the *shortest* feasible path. On subpath $[s \dots u]$, the RCFP search proceeds as in case (1).
- *P contains multiple charging stops:* Let u and u' be two consecutive charging stations on P, which divide P into subpaths $[s \dots u]$, $[u \dots u']$ and $[u' \dots t]$. Lemma 2 in [8] shows that for CFP, the *optimal* departure time at u always corresponds to charging to either $\text{in}_{[u \dots u']}$ or to a breakpoint of cf_u. Similarly, after the RCFP search reaches u, the departure time at u' always corresponds to charging to either $\text{in}_{[u \dots u']}$, or to a breakpoint of $\text{cf}_{u'}^{-1}$, which is optimal.

Next, we show that the minimum time label in RCFP search is not dominated by other labels and reaches s the first. The first claim follows from the dominance criterion for RCFP, which is symmetric to that of CFP: A label ℓ_v is dominated if it results in a higher total travel time for every possible initial SoC at v. This implies that a dominated label can not result in a unique optimal solution, since replacing the sub-path to the target it represents with the sub-path of the label dominating it would result in a better or equal solution. Lastly, since labels are ordered by travel time at all $L_{uns}(v)$, the label with minimum total travel time reaches s first. ◀

4.1.1 Computing SCM with Reverse CFP

If the Reverse CFP algorithm does not terminate when the search reaches s and is instead allowed to continue to run until PQ is empty, we would have the set of all pareto-optimal feasible paths from s to t at s. This set of pareto-optimal feasible paths forms the Starting Charge Map between vertices s and t.

▶ **Theorem 5.** *If the RCFP algorithm is run from $t \in V$ with $\beta_t = 0$ until the priority queue is empty, the Pareto-set of labels at every $s \in V$ is equivalent to SCM_{st}.*

Proof. We prove Theorem 5 by showing that after running the RCFP from t, a starting SoC β_s, the label set at s contains a label that corresponds to $SCM_{st}(\beta_s)$. For this, we add temporary *virtual* vertices $s^{(M-x)}$ and an edge from $s^{(M-x)}$ to s with energy consumption x to the network, as depicted in Figure 2. From Lemma 4, we know that RCFP can compute a shortest feasible path P from $s^{(M-x)}$ to t. Note that by construction, P must contain s and $\text{in}_{[s\ldots t]} \leq x$, since $(M - x)$ energy is consumed on the edge from $s^{(M-x)}$ to s. Thus, a label ℓ must exist at s that represents the shortest feasible path from s to t and requires an initial SoC of at most $x.\ell$ corresponds to $SCM_{st}(x)$. Since the computation of RCFP in the network without $s^{(M-x)}$ is independent of the existence of $s^{(M-x)}$, the RCFP algorithm has to compute the label ℓ before the priority queue runs empty even if $s^{(M-x)}$ is not part of the network. ◀

5 Buffer Maps

Like Starting Charge Maps, a Buffer Map is a generalization of the Shortest Feasible Path Problem; albeit instead of unknown starting charge β_s, the lower bound of minimum allowed SoC along the path is raised from 0 to an arbitrary $\mathfrak{b} \in [0, M]$. Formally,

▶ **Definition 6.** *A* buffer map $BM_{st} : [0, M] \to P$ *between a source $s \in V$ and target $t \in V$ is a function that maps a given* buffer SoC $\mathfrak{b} \in [0, M]$ *to the corresponding shortest feasible path P such that the EV maintains at least $\mathfrak{b}$ SoC at all points in P.*

Further, like SCMs, Buffer Maps can be used to show alternative routes to EV drivers who can decide upon the degree of acceptable stranding risk along the route. However, a key difference between the two abstractions and their usage is that while SCMs are used to get alternative routes depending on the *starting state* of the EV, alternative routes in buffer maps differ on the basis of *projected EV behaviour along the route*. In this way, alternative routes in BMs offer strong guarantees against stranding risk for EV drivers; not surprisingly, they are also more expensive to compute. Note that this problem would also qualify as what is referred to as "profile query" in the literature, since we ask for an optimal solution for arbitrary initial SoC [10, 13]. However, unlike [10, 13], we consider a multi-criteria variant of this problem and also allow intermediate charging stops.

For each distinct $\mathfrak{b}$, a run of the CFP algorithm can yield a shortest feasible path with the minimum SoC equal to $\mathfrak{b}$. A brute force approach to computing a Buffer Map is to run CFP several times, setting $\mathfrak{b}$ to each value in $[0, M]$. However, this approach is not feasible since the interval $[0, M]$ contains infinite values. In the next subsection, we present an exact, practical algorithm to compute a buffer map.

5.1 Iterative Charging Function Propagation

Each EV path consists of a sequence of legs. We define:

▶ **Definition 7.** *Given an SoC $\mathfrak{b} \in [0, M]$, a* critical leg *of a shortest feasible path P is one on which the SoC drops to $\mathfrak{b}$.*

CFP computes the exact amount of charge that an EV charges at every station along a feasible route P in order to minimize total travel time. However, to ensure that the minimum SoC of the EV along P never drops below a given $\mathfrak{b} \in [0, M]$, the EV must charge extra on

the charging stations adjacent to critical legs along P. The amount of extra energy to charge at such stations is exactly equal to that required to maintain at least $\mathfrak{b}$ SoC along the route, and is called the *buffer energy*.

Our approach to computing a Buffer Map BM_{st} for a given source $s \in V$ and target $t \in V$ works in *iterations*. Every iteration starts with choosing a value $\mathfrak{b}' \in [0, M]$. An augmented variant of CFP is run that returns a shortest feasible path P' such that the minimum SoC of the EV along P' is equal to $\mathfrak{b}'$. A collection of all such P' constitutes the set of paths in BM_{st}. Therefore, our approach has two components: first, an augmented variant of the CFP that respects the buffer SoC $\mathfrak{b}'$, and second, an algorithm that computes the increase in $\mathfrak{b}'$ on every iteration.

5.1.1 Augmenting CFP

The first iteration of our algorithm starts with $\mathfrak{b}' = 0$. The augmented CFP search starts from s with an SoC β_s and propagates towards t. Assume that the search requires charging at consecutive stations u' and u, and reaches $v \in V$. Let the breakpoints of $\text{cf}_{u'}$ be $[B^1_{u'} B^2_{u'} \dots B^m_{u'}]$, where $B^i_{u'} = (\tau^i_{u'}, SoC^i_{u'}), 1 \leq i \leq m$ where SoC'_i is EV's resultant SoC after charging for time $\tau^i_{u'}$. Similarly, the breakpoints of cf_u are $[B^1_u B^2_u ... B^n_u]$.

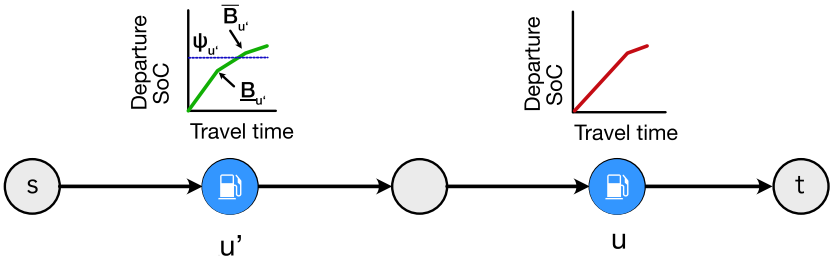


Figure 3 Augmented CFP setup.

Recall that CFP sets the amount of charge added to the EV at a station only after the search reaches the next charging station. Let the EV's SoC be $\psi_{u'}$ at departure after charging at station u'. Also, let $\underline{B}_{u'} = (\underline{\tau_{u'}}, \underline{SoC_{u'}})$ be the breakpoint of $\text{cf}_{u'}$ with SoC immediately lesser or equal to $\psi_{u'}$, and $\overline{B}_{u'} = (\overline{\tau_{u'}}, \overline{SoC_{u'}})$ be the next breakpoint after $\underline{B}_{u'}$. Therefore, $\underline{SoC}_{u'} \leq \psi_{u'} < \overline{SoC}_{u'}$. Figure 3 shows an example of the $\underline{B}_{u'}$ and $\overline{B}_{u'}$ corresponding to a given $\psi_{u'}$. Similarly, given cf_u and ψ_u, $\underline{SoC}_u \leq \psi_u \leq \overline{SoC}_u$.

At $v \in V$, a label of the search is given by $\mathfrak{l} = \langle \tau_v, \beta_u, u, f_{[u...v]}, \rho_v, \delta_v \rangle$, where τ_t, β_u, u, and $f_{[u...v]}$ are analogous to regular CFP, ρ_v is the time required to add unit buffer energy to the EV on the current path, and δ_v is the maximum SoC up to which it can be charged without a loss in charging rate (due to concavity of charging functions).

▶ **Lemma 8.** *Let P be a shortest feasible $s-t$ path with k charging stops on P and $\mathfrak{b}$ be the minimum allowed SoC along P. Assume that the EV arrives at i^{th} charging station with SoC α_i, charges for t_i time, and departs with SoC ψ_i. Further, let C be the charging stations at the beginning of critical legs in P. To increase the buffer energy along P by ϵ, increasing departure SoC ψ_i to $(\psi_i + \epsilon)$ on all stations in C is an optimal solution if:*

(1) *On charging stations in C, $f(\psi_i + \epsilon) - f(\psi_i) = \epsilon$, i.e. charging ϵ more increases the residual SoC at t by ϵ.*

(2) *On all non-critical legs, the minimum allowed SoC is at least $\mathfrak{b} + \epsilon$.*

(3) *For all charging stations in C, the charging function is differentiable and does not have breakpoints with SoCs in range $[\psi_i, (\psi_i + \epsilon)]$.*

(4) *For all charging stations at the end of a critical leg, the charging function is differentiable and does not have a breakpoint in SoC range $[\alpha_i, (\alpha_i + \epsilon)]$.*

Proof. First, note that increasing ψ_i at all charging stations in C by ϵ is sufficient to increase the total buffer by ϵ – this follows immediately from conditions (1) and (2).

Let a *solution* S be the set of charging stops and charging times along path P, resulting from an Augmented CFP run between vertices s to t. We will show that no other solution can result in a lower total travel time along path P without changing at least one edge in P. Assume for contradiction, a solution S' has a lower total travel time than S along same path P. In order to increase the buffer energy for S by ϵ, ψ_i for each charging station in C must be increased by at least $(\psi_i + \epsilon)$. This can only be achieved by charging additional energy at a station on P.

Let j be the charging stop closest to t at which charging time differs between S and S'. We claim that there must be a critical leg after departing from j and that its departure SoC is $(\psi_j + \epsilon)$ – if this were not the case, we could decrease the departure SoC at j to $(\psi_j + \epsilon)$, which would be sufficient to increase buffer energy by ϵ, giving us a faster solution and contradicting the assumption that S is optimal. This implies that we can decrease the departure SoC at j to $(\psi_j + \epsilon)$, which is sufficient to increase buffer energy by ϵ, which gives us a faster solution contradicting the assumption that S is optimal. Since the departure SoC on j is equal to $(\psi_j + \epsilon)$, the arrival SoC at j must be greater. In other words, we charge more at some other stop i so we can charge less at j. But then, we can create a faster solution for buffer energy $\mathfrak{b}$ as follows: there exists a $\delta > 0$ such that we can charge δ more at i and charge δ less at j (since the charging function is concave and differentiable around ψ_j, the charging rate remains the same as for $(\psi_j + \epsilon)$). This contradicts the fact the solution S for buffer SoC $\mathfrak{b}$ was optimal. ◀

The CFP search starts from s with $\rho_s = 0$ and $\delta_s = M$. Assume that the search reaches charging station u after charging at a prior station u'. Let $\mathfrak{l}_u = \langle \tau_v, \beta_u, u, f_{[u \ldots v]}, \rho_v, \delta_v \rangle$ be the current label extracted from priority queue. If $\psi_{u'} = \mathfrak{b}'$, i.e. the leg $[u' \ldots u]$ is a critical leg, we set $\delta_u = \min(\delta_{u'}, SoC_{B_2} - SoC_{B_1}, \overline{SoC}_u - \mathrm{cf}_u(f_{[u' \ldots u]}(\psi_{u'})))$, where SoC_{B_1} and SoC_{B_2} are the SoC of the first and second breakpoints of the SoC function of $\mathfrak{l}_u$. We also set $\rho_u = \rho_{u'} + \frac{(\overline{\tau_{u'}} - \underline{\tau_{u'}})}{(\overline{SoC_{u'}} - \underline{SoC_{u'}})} - \frac{(\overline{\tau_u} - \underline{\tau_u})}{(\overline{SoC_u} - \underline{SoC_u})}$. If $[u' \ldots u]$ is not a critical leg, the $\delta_u = \min(\delta_{u'}, \mathrm{in}_{[u' \ldots u]} - \mathfrak{b}')$, and $\rho_u = \rho_{u'}$.

A label $\mathfrak{l}_u$ dominates $\mathfrak{l}'_u$ if the SoC function of $\mathfrak{l}_u$ dominates the SoC function of $\mathfrak{l}_{u'}$, and $\rho_u \leq \rho_{u'}$. In other words, a label $\mathfrak{l}$ dominates $\mathfrak{l}'$ if it represents a faster path to which the buffer energy can be added at a faster rate.

5.1.2 Computing $\mathfrak{b}'$ for the next iteration

On an iteration, we let the augmented CFP run and collect the complete set of non-dominated labels at target t. Let the set of labels collected at t be $\mathfrak{L}$. The Augmented CFP search guarantees that every $\mathfrak{l}_i \in \mathfrak{L}$ represents a feasible path where the SoC along the path does not drop below $\mathfrak{b}$. Let the label $\mathfrak{l}_{min}$ have the minimum total travel time of all $\mathfrak{l} \in \mathfrak{L}$. Next, we need to determine the maximum buffer energy that can be added at stations adjacent to critical legs of the shortest feasible path P found in the current iteration, while ensuring that no other feasible path becomes a better (faster) choice than P. We can solve this problem geometrically on an X-Y plane, where X and Y axis represent the buffer SoC and the total travel time of the EV respectively.

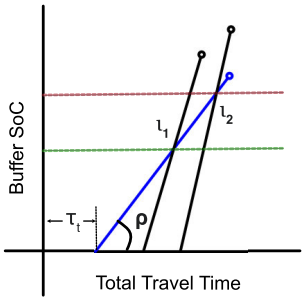


Figure 4 Each line on the X-Y plane represents a label $\mathfrak{l}_i \in \mathfrak{L}$ for iteration N. Highlighted blue *label line segment* represents the minimum time label $\mathfrak{l}_{min}$. The slope of blue label line segment is $\rho \in \mathfrak{l}_{min}$, and the X-intercept is equal to travel time $\tau_t \in \mathfrak{l}_{min}$. It intersects with two other label line segments at ι_1 and ι_2. Similarly, let intersection points be $\{\iota_1, \iota_2, ..., \iota_n\}$ if $\mathfrak{L}$ contains more labels. b' for the next iteration is equal to the minimum buffer SoC in $\{\iota_1, \iota_2, ..., \iota_n\}$ (SoC of shown green line).

For a label $\mathfrak{l}_t$, we draw a *label line segment* with slope $\rho_t \in \mathfrak{l}_t$, and the X-intercept equal to the total travel time τ_t of $\mathfrak{l}$. Further, the maximum ordinate of the line segment is given by $\delta_t \in \mathfrak{l}_t$. Figure 4 shows an example where $\mathfrak{L}$ contains three labels. The next step is to find the *globally minimum buffer SoC*, δ_{min} to which the EV can be charged the fastest among all labels in $\mathfrak{L}$. To find such a value, we start with the label line segment for $\mathfrak{l}_{min}$, and find its intersections with all other label line segments on the plane. Let the set of such intersections be $\{\iota_1, \iota_2, ..., \iota_n\}$. Since Figure 4 has only three label line segments, it shows two intersection points ι_1 and ι_2. Thus, δ_{min} is given by the buffer SoC of the intersection point that lowest on the Y-axis in the plane. For the next iteration, we set $\mathfrak{b}' = \delta_{min}$ and add the feasible path represented by $\mathfrak{l}_t$ to the buffer map BM.

▶ **Lemma 9.** *The global delta selection algorithm is correct, i.e. no feasible path has a lower total travel time and can add buffer energy faster than the chosen route given by the algorithm.*

Proof. We prove geometrically. Since all charging functions are convex with a positive slope, the slopes of all label line segments in the X-Y plane are positive. Further, since $\mathfrak{l}_{min}$ has the smallest X-intercept, in buffer SoC interval $[0, \text{SoC of } \iota_1]$, no other label in $\mathfrak{L}$ can charge the EV to a higher buffer SoC in lesser time. ◀

As we increase $\mathfrak{b}'$ on each iteration, the augmented CFP search becomes more selective and the number of feasible paths from s to t decreases, since only on fewer paths would an EV be able to maintain a higher minimum SoC. The iterations terminate when $\mathfrak{b}'$ becomes high enough so the CFP search does not return any feasible paths.

▶ **Theorem 10.** *The Iterative CFP algorithm terminates and computes BM_{st} correctly.*

Proof. We have already argued that we compute ρ and δ correctly for labels propagated by the Augmented CFP search, and that for label $\mathfrak{l}$ it gives us the minimum additional required charging time in order to increase the buffer energy by any value in $[0, \delta]$ on the feasible path represented by $\mathfrak{l}$. We now show that the solutions added to the buffer map are indeed optimal and there is no remaining path with a shorter time for some value of buffer energy. Assume for contradiction, that we add a label $\mathfrak{l}$ to the buffer map, for which there exists a label $\mathfrak{l}'$ that offers a faster solution for some buffer energy. Observe that this implies that it is not a part of the Pareto set at the target, since the global delta computation finds the best label in that set by lemma 9. We can now distinguish two cases:

1. $\mathfrak{l}'$ *represents a feasible path with at least one critical leg:* Since $\mathfrak{l}'$ can not have a faster (minimum) traversal time than $\mathfrak{l}$ by construction (the algorithm selected $\mathfrak{l}$ and added it to the buffer map because it is the label with minimum travel time), it can only become the better solution after adding additional charge so it yields shorter total travel time for higher buffer energy. In other words, $\mathfrak{l}'$ offers a better charging rate and therefore is not dominated by $\mathfrak{l}$, which implies that it (or another dominating label) must be a member of the Pareto set. This must result in a lower intersection point on the Y-axis than δ during the global delta computation, which contradicts our assumption.
2. $\mathfrak{l}'$ *represents a feasible path with no critical leg:* This implies it has no charging stop (if there was a label with a charging stop but no critical leg, we could always charge less to obtain a faster solution). This means it cannot be dominated by $\mathfrak{l}$ because it has $\rho = 0$, and therefore it or another single-leg path must be a part of the Pareto set, which implies that it is taken into account when computing the global value of δ, again leading to a contradiction. ◀

Several factors can affect the total number of iterations required to compute BM: the distance between s and t, the total number of charging stations required to reach from s to t, which in turn depends on the parameters of the EV under consideration. The number of iterations further depends on the number of breakpoints in charging functions along the feasible paths from s to t. However, in practice, the number of iterations remains small for the following reasons: First, $\mathrm{cf}_u, u \in S$ are usually simple, linear functions up to 80% charge and only have a small number of breakpoints in the $80 - 100\%$ range. Next, most EV trips tend to not have a large number of charging stops along the way, and as EV ranges increase, this number would further decrease.

6 Experiments

We implemented our algorithms in C++ using Apple clang version 10.0.1 with $-O3$ optimizations. All experiments were run on macOS 10.14.6 using a Mac Pro 6,1 with a quad-core Intel Xeon E5 (3.7 GHz base clock). The processor has 256 KB of per-core L2 and 10 MB of shared L3 cache. The machine has 64 GBs of DDR3-ECC memory clocked at 1866 MHz.

6.1 Preparing a realistic EV Routing instance

Table 1 Our road network is taken from OpenStreetMap, public charging stations data from the Alternative Fuel Data Center [2], elevations from NASADEM [38] and an energy consumption model from a Nissan Leaf 2013 [20].

Dataset	Vertices	Edges	|Ch. stations|
Oregon (contracted)	502327	710107	323
California (contracted)	2547618	3741891	1406

We extract the road networks of Oregon and California from OpenStreetMap (OSM)[1] and label each edge with travel time equal to geographic distance divided by the maximum allowed speed for the road segment type. We contract all vertices with degrees ≤ 2 for our experiments, keeping only the largest connected component of the network. Table 1 shows the size of road networks after contraction.

[1] `https://openstreetmap.org/`

Next, we add the elevation to each vertex of the network, taken by sampling the NASADEM elevation dataset at $30m$ resolution [38]. The elevation is required to compute the energy consumption on every edge of the network, which we derive from a microscopic EV energy consumption model for a Nissan Leaf 2013 [20].

Lastly, we extract the locations of public EV charging stations in Oregon and California from the Alternative Fuels Data Center [2]. For each charging station in the dataset, we mark the vertex geographically closest to it as the charging station. We assign each charging station vertex one of three charging functions: i) a *slow* linear function that charges the EV to full battery in 120 minutes; ii) a *fast* charging function that charges the EV to 80% in 30 minutes and to full capacity in 60 minutes, and iii) a *fastest* charging function that charges to 80% capacity in 20 minutes, and to full in 40 minutes. We arbitrarily assign 60% of all charging stations the *slow* charging function, another 30% stations the *fast*, and the remaining 10% the *fastest* charging functions.

To allow for tests with reasonable running times, we make it easier for a label to dominate another in the (Reverse) CFP search. We do this by adding a constant *slack energy consumption* ϵ to the dominance criterion in all three algorithms. Given labels ℓ_1 and ℓ_2, ℓ_1 dominates ℓ_2 iff all breakpoints of ℓ_1's SoC function have a higher energy than breakpoints of ℓ_2's SoC function after decreasing each breakpoint by ϵ energy. We set ϵ to 1% of the total battery capacity of the EV. Similar modifications to the dominance criteria have been proposed in earlier work, e.g. see [5, 12].

6.2 Reverse Shortest Feasible Path Queries & Starting Charge Maps

Table 2 Average performance of 1000 queries running RCFP vs. variants of standard CFP. The EV is always assumed to start with 100% SoC at source. CFP with stopping criterion terminates after finding only one feasible route, and is therefore much faster than regular CFP which returns all feasible routes. RCFP can be seen to perform at par with CFP without stopping criterion. Time shown in seconds, also shown – no. of labels extracted from priority queue, alternative routes to t, and the no. of times search reached target. Targets found differ between RCFP and CFP because of the difference in dominance criteria.

		16 kWh				**32 kWh**			
	Alg.	Time	kLabels	|Routes|	Targets	Time	kLabels	|Routes|	Targets
Oregon	CFP (Stp)	1.767	933	0.709	709	1.510	873	0.895	895
	CFP	3.853	1758	4.962	709	3.629	1973	5.634	895
	RCFP	4.861	2477	7.703	710	3.502	2370	7.176	895
California	CFP (Stp)	34.847	10141	0.722	722	21.805	8645	1.0	1000
	CFP	70.076	19684	8.88	722	61.171	21596	9.837	1000
	RCFP	66.096	22571	11.467	724	46.617	22191	11.645	1000
		64 kWh				**128 kWh**			
		Time	kLabels	|Routes|	Targets	Time	kLabels	|Routes|	Targets
Oregon	CFP (Stp)	0.877	730	1.0	1000	0.678	621	1.0	1000
	CFP	2.648	1859	5.205	1000	2.521	1751	5.04	1000
	RCFP	2.641	2071	6.599	1000	2.241	1877	5.76	1000
California	CFP (Stp)	13.919	6631	1.0	1000	7.221	5006	1.0	1000
	CFP	47.197	18273	8.429	1000	25.182	13752	6.304	1000
	RCFP	36.568	19079	9.897	1000	16.255	12220	6.563	1000

Table 2 shows the results of running 1000 SFP and RSFP queries with several standard EV battery capacities (16, 32, 64, and 128 kWh) between random vertices in the road networks of Oregon and California. The table compares the performance of three algorithms–forward

CFP with *stopping criterion*, which makes the search terminates as soon as it reaches t; *full* forward CFP that runs till all pareto-optimal feasible paths from s to t are found; and the Reverse CFP algorithm as presented in Section 4.

We find that the CFP with stopping criterion performs at least a factor of two faster than full CFP that computes the pareto-optimal set of feasible paths. This is hardly surprising as the full CFP offers a richer set of routes which planners can use, in lieu of more computational overhead. However, if faster queries are desirable at the cost of alternative routes, the same technique can be applied to the reverse CFP algorithm with little effort. Target pruning [9] is another closely related technique that can achieve the same goal.

We observe that for both networks, SFP and RSFP query times generally decrease with increase in range of the EV, with a notable exception of capacity increase from 16 to 32 kWh, in which case the reverse search query times increase for the Oregon network and full CFP query times for the California network. This can be explained as follows: As the battery capacity increases, the (R)CFP search is able to reach vertices farther away. However, with increase in range, the slack energy ϵ also increases, making it easier for a label to dominate another, so fewer labels are settled in the search. The net effect of the two opposing factors, in this case, is that the total query time increases.

6.3 Iterative CFP and Buffer Maps

Table 3 Average performance of Iterative CFP to answer 1000 Buffer Map queries (with 50 and 100% starting SoC) between random vertices on the Oregon road network.

	Range	Time (s)	kLabels	Iterations	Avg. $\lvert BM \rvert$	Targets
50%	16 kWh	67.405	30754	7.03	6.103	640
	32 kWh	99.310	45685	9.987	9.061	878
	64 kWh	32.571	25441	9.37	8.538	1000
	128 kWh	17.440	15316	7.013	6.359	1000
100%	16 kWh	198.395	57545	10.573	9.57	709
	32 kWh	85.484	47106	14.285	13.29	895
	64 kWh	53.706	37930	14.225	14.22	1000
	128 kWh	14.240	16183	10.611	9.635	1000

Table 3 shows the results of 1000 Iterative CFP queries between random vertices in the Oregon network. We do not report the running times for California, since they were found to be impractical with some queries running for more than 3 hours.

The total running time of the Iterative CFP algorithm has two components: The cost of Augmented CFP runs and the cost of computing the minimum global δ energy in each round. The cost of global delta computation is negligible in practice, since the number of Augmented CFP labels reaching the target vertex is often low. In Table 3, note that an Augmented CFP run takes longer than full CFP. This is caused due to inclusion of an additional parameter (charging rate) in the dominance criteria of the Augmented CFP.

The Iterative CFP is slower than the other algorithms discussed. This is expected as the algorithm involves running several iterations of an exponential-time shortest path computation. Our networks do not use standard speedup techniques like Contraction Hierarchies (CHs) [28], their multicriteria variant [27], or CRP [15, 16], though. Applying any of these techniques can significantly reduce query times by reducing the number of vertices explored to find shortest paths. A combination of speedup techniques such as CHs and A* search could be further applied for even greater speedups [6] at the cost of additional complexity.

7 Conclusion and Future Work

In this paper, we introduced Starting Charge Maps and Buffer Maps, which are helpful in preventing EV users' range anxiety and enable other use cases (such as minimizing trip time by charging more at home). Both problems require extending the known Shortest Feasible Path problem, essentially increasing its output by another dimension. Similar to profile queries in time-dependent route planning [11], this requires more sophisticated algorithms for Buffer Maps, which is reflected in the running times we observed in our experimental evaluation. For Starting Charge Maps, however, we proposed a simple and elegant approach which is in large parts symmetric to the known CFP algorithm and, as a result, computes them with similar running times, as out experimental results confirm.

Possible future work includes (heuristic) improvements of the Buffer Map search, or integration with A* and CH for faster queries as done by the CHArge algorithm [7, 8]. We may further consider related problem settings such having the SoC buffer dependent on the distance between charging stops.

References

1 Yazan Al-Wreikat, Clara Serrano, and José Ricardo Sodré. Driving behaviour and trip condition effects on the energy consumption of an electric vehicle under real-world driving. *Appl. Energy*, 297:117096, September 2021.

2 Alternative Fuels Data Center. Electric vehicle charging station locations. `https://afdc.energy.gov/fuels/electricity_locations.html`, 2021. Accessed: 2021-6-9.

3 Andreas Artmeier, Julian Haselmayr, Martin Leucker, and Martin Sachenbacher. The shortest path problem revisited: Optimal routing for electric vehicles. In *KI 2010: Advances in Artificial Intelligence*, Lecture Notes in Computer Science, pages 309–316. Springer, Berlin, Heidelberg, September 2010.

4 Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm Engineering*, Lecture Notes in Computer Science, pages 19–80. Springer, Cham, 2016.

5 Lucas S Batista, Felipe Campelo, Frederico G Guimarães, and Jaime A Ramírez. A comparison of dominance criteria in many-objective optimization problems. In *2011 IEEE Congress of Evolutionary Computation (CEC)*, pages 2359–2366, June 2011.

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. *Algorithms*, 2008.

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 SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 44. ACM, November 2015.

8 Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. *Transportation Science*, 53(6):1627–1655, November 2019.

9 Moritz Baum, Julian Dibbelt, Lorenz Hübschle-Schneider, Thomas Pajor, and Dorothea Wagner. Speed-Consumption tradeoff for electric vehicle route planning. In *14th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems*, OpenAccess Series in Informatics (OASIcs), pages 138–151. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2014.

10 Moritz Baum, Julian Dibbelt, Thomas Pajor, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Energy-Optimal routes for battery electric vehicles. *Algorithmica*, December 2019.

11 Moritz Baum, Julian Dibbelt, Thomas Pajor, and Dorothea Wagner. Dynamic Time-Dependent route planning in road networks with user preferences. In *Experimental Algorithms*, volume 9685 of *Lecture Notes in Computer Science*, pages 33–49, Cham, 2016. Springer International Publishing.

12 Moritz Baum, Julian Dibbelt, Dorothea Wagner, and Tobias Zündorf. Modeling and engineering constrained shortest path algorithms for battery electric vehicles. In *LIPIcs-Leibniz International Proceedings in Informatics*, volume 87, 2017.

13 Moritz Baum, Jonas Sauer, Dorothea Wagner, and Tobias Zündorf. Consumption profiles in route planning for electric vehicles: Theory and applications. In *16th International Symposium on Experimental Algorithms (SEA 2017)*. drops.dagstuhl.de, 2017.

14 Cedric De Cauwer, Wouter Verbeke, Thierry Coosemans, Saphir Faid, and Joeri Van Mierlo. A Data-Driven method for energy consumption prediction and Energy-Efficient routing of electric vehicles in Real-World conditions. *Energies*, 10(5):608, May 2017.

15 Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. Customizable route planning. In *Experimental Algorithms*, pages 376–387. Springer, Berlin, Heidelberg, May 2011.

16 Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. Customizable route planning in road networks. *Transportation Science*, 51(2):566–591, May 2015.

17 Daniel Delling and Dorothea Wagner. Time-Dependent route planning. In *Robust and Online Large-Scale Optimization*, Lecture Notes in Computer Science, pages 207–230. Springer, Berlin, Heidelberg, 2009.

18 Matthias Eisel, Ilja Nastjuk, and Lutz M Kolbe. Understanding the influence of in-vehicle information systems on range stress – insights from an electric vehicle field experiment. *Transp. Res. Part F Traffic Psychol. Behav.*, 43:199–211, November 2016.

19 Jochen Eisner, Stefan Funke, and Sabine Storandt. Optimal route planning for electric vehicles in large networks. *AAAI*, 25(1), August 2011.

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

21 Chiara Fiori, Vittorio Marzano, Vincenzo Punzo, and Marcello Montanino. Energy consumption modeling in presence of uncertainty. *IEEE Trans. Intell. Transp. Syst.*, pages 1–12, 2020.

22 Matthew William Fontana. *Optimal routes for electric vehicles facing uncertainty, congestion, and energy constraints*. PhD thesis, Massachusetts Institute of Technology, 2013.

23 Luca Foschini, John Hershberger, and Subhash Suri. On the complexity of Time-Dependent shortest paths. *Algorithmica*, 68(4):1075–1097, April 2014.

24 Thomas Franke and Josef F Krems. Interacting with limited mobility resources: Psychological range levels in electric vehicle use. *Transp. Res. Part A: Policy Pract.*, 48:109–122, February 2013.

25 Thomas Franke, Isabel Neumann, Franziska Bühler, Peter Cocron, and Josef F Krems. Experiencing range in an electric vehicle: Understanding psychological barriers: Experiencing range. *Appl. Psychol.*, 61(3):368–391, July 2012.

26 Thomas Franke, Nadine Rauh, Madlen Günther, Maria Trantow, and Josef F Krems. Which factors can protect against range stress in everyday usage of battery electric vehicles? toward enhancing sustainability of electric mobility systems. *Hum. Factors*, 58(1):13–26, February 2016.

27 Stefan Funke and Sabine Storandt. Polynomial-time construction of contraction hierarchies for multi-criteria objectives. In *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, pages 41–54, Philadelphia, PA, USA, 2013. Society for Industrial and Applied Mathematics.

28 Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, August 2012.

29 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*, pages 193–202. ACM, November 2014.

30 Gerhard Huber, Klaus Bogenberger, and Hans van Lint. Optimization of charging strategies for battery electric vehicles under uncertainty. *IEEE Trans. Intell. Transp. Syst.*, pages 1–17, 2020.

31 Malte F Jung, David Sirkin, Turgut M Gür, and Martin Steinert. Displayed uncertainty improves driving experience and behavior: The case of range anxiety in an electric car. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2201–2210. ACM, April 2015.

32 Johannes Kester. Security in transition(s): The low-level security politics of electric vehicle range anxiety. *Security Dialogue*, 50(6):547–563, December 2019.

33 Yan Li, Pratik Kotwal, Pengyue Wang, Yiqun Xie, Shashi Shekhar, and William Northrop. Physics-guided energy-efficient path selection using on-board diagnostics data. *ACM/IMS Trans. Data Sci.*, 1(3):1–28, September 2020.

34 Yan Li, Shashi Shekhar, Pengyue Wang, and William Northrop. Physics-guided energy-efficient path selection: a summary of results. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, November 2018.

35 Ernesto Queirós Vieira Martins. On a multicriteria shortest path problem. *Eur. J. Oper. Res.*, 16(2):236–245, May 1984.

36 Michail Masikos, Konstantinos Demestichas, Evgenia Adamopoulou, and Michael Theologou. Energy-efficient routing based on vehicular consumption predictions of a mesoscopic learning model. *Appl. Soft Comput.*, 28:114–124, March 2015.

37 Sören Merting, Christian Schwan, and Martin Strehler. Routing of electric vehicles: Constrained shortest path problems with resource recovering nodes. In *OASIcs-OpenAccess Series in Informatics*, volume 48. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik GmbH, Wadern/Saarbruecken, Germany, 2015.

38 J P L Nasa. NASADEM merged DEM global 1 arc second V001 [dataset]. `http://dx.doi.org/10.5067/MEaSUREs/NASADEM/NASADEM_HGT.001`, 2020. Accessed: 2021-6-9.

39 Zachary A Needell, James McNerney, Michael T Chang, and Jessika E Trancik. Potential for widespread electrification of personal vehicle travel in the united states. *Nature Energy*, 1:16112, August 2016.

40 Dario Pevec, Jurica Babic, Arthur Carvalho, Yashar Ghiassi-Farrokhfal, Wolfgang Ketter, and Vedran Podobnik. Electric vehicle range anxiety: An obstacle for the personal transportation (r)evolution? In *2019 4th International Conference on Smart and Sustainable Technologies (SpliTech)*, pages 1–8, June 2019.

41 Dario Pevec, Jurica Babic, Arthur Carvalho, Yashar Ghiassi-Farrokhfal, Wolfgang Ketter, and Vedran Podobnik. A survey-based assessment of how existing and potential electric vehicle owners perceive range anxiety. *J. Clean. Prod.*, 276:122779, December 2020.

42 Xuewei Qi, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J Barth. Data-driven decomposition analysis and estimation of link-level electric vehicle energy consumption under real-world traffic conditions. *Transp. Res. Part D: Trans. Environ.*, 64:36–52, October 2018.

43 Payas Rajan and Chinya V Ravishankar. The phase abstraction for estimating energy consumption and travel times for electric vehicle route planning. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '19, pages 556–559, New York, NY, USA, 2019. ACM.

44 Nadine Rauh, Thomas Franke, and Josef F Krems. User experience with electric vehicles while driving in a critical range situation – a qualitative approach. *IET Intel. Transport Syst.*, 9(7):734–739, July 2015.

45 Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. Efficient energy-optimal routing for electric vehicles. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, AAAI'11, pages 1402–1407. AAAI Press, August 2011.

46 Stefan Sautermeister, Max Falk, Bernard Bäker, Frank Gauterin, and Moritz Vaillant. Influence of measurement and prediction uncertainties on range estimation for electric vehicles. *IEEE Trans. Intell. Transp. Syst.*, 19(8):2615–2626, August 2018.

47 René Schönfelder and Martin Leucker. Abstract routing models and abstractions in the context of vehicle routing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, June 2015.

48 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, New York, NY, USA, 2012. ACM.

49 Sabine Storandt and Stefan Funke. Cruising with a Battery-Powered vehicle and not getting stranded. In *AAAI*, volume 3, page 46, 2012.

50 Martin Strehler, Sören Merting, and Christian Schwan. Energy-efficient shortest routes for electric and hybrid vehicles. *Trans. Res. Part B: Methodol.*, 103(Supplement C):111–135, September 2017.