Cruising with a Battery-Powered Vehicle and not Getting Stranded

Sabine Storandt and **Stefan Funke**
Universität Stuttgart, Institut für Formale Methoden der Informatik, 70569 Stuttgart, Germany
storandt, funke@fmi.uni-stuttgart.de

Abstract

The main hindrance to a widespread market penetration of battery-powered electric vehicles (BEVs) has been their limited energy reservoir resulting in cruising ranges of few hundred kilometers unless one allows for recharging or switching of depleted batteries during a trip. Unfortunately, recharging typically takes several hours and battery switch stations providing fully recharged batteries are still quite rare – certainly not as widespread as ordinary gas stations. For not getting stranded with an empty battery, going on a BEV trip requires some planning ahead taking into account energy characteristics of the BEV as well as available battery switch stations. In this paper we consider very basic, yet fundamental problems for E-Mobility: *Can I get from A to B and back with my BEV without recharging in between? Can I get from A to B when allowed to recharge? How can I minimize the number of battery switches when going from A to B?* We provide efficient and mathematically sound algorithms for these problems that allow for the energy-aware planning of trips.

Introduction

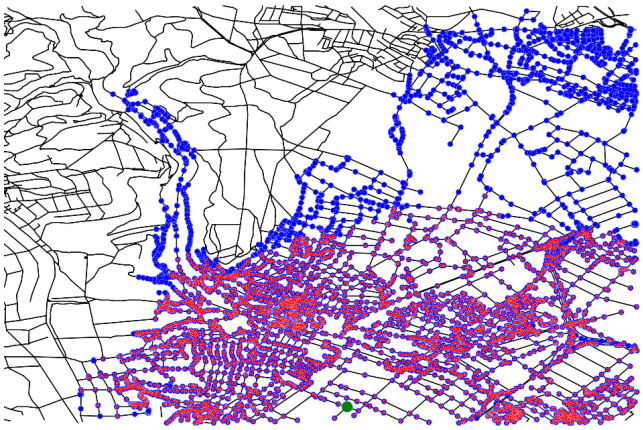

Figure 1: Ev-reachable nodes (blue) and strongly ev-connected nodes (red) for a given source node (green).

In recent years E-mobility has been identified as important means to reduce the consumption of fossil fuels. Governments offer reduced taxes for electric vehicles (EVs) compared to fuel driven cars, and provide federal funding for the development of green technologies. Nevertheless EVs still wait for their great breakthrough. One reason being that most EVs are battery powered, which limits their cruising range. On the other hand, BEVs feature advantages like the ability to recuperate energy during deceleration phases or when going downhill. Of course, this does not suffice for long trips, hence a network of charging stations needs to be established. Denmark is a pioneer in this field, not only providing power for free at several parking lots but also charging stations, where the batteries can be exchanged in order to make recharging not more time-consuming than refueling – so called *battery switch stations (BSS)*. Until a dense network of charging stations has been established, it is of utmost importance to take energy constraints into account when planning routes with a BEV. Algorithms for energy-efficient routing of BEVs have only recently become the focus of active research (Artmeier et al. 2010; Eisner, Funke, and Storandt 2011; Sachenbacher et al. 2011), many important questions are still to be settled. In this paper we consider very basic, yet important problems in this context: *Can I get from A to B and back with my BEV without recharging in between? Can I get from A to B when allowed to switch batteries on the way? How can I minimize the number of battery switches when going from A to B?* Only if such fundamental questions are answered reliably and efficiently, we can hope for a fast transition to E-mobility. For example, in the pilot project *E-Tour*[1] funded by the German Ministry of Economics and Technology, tourists can use BEVs to explore the Bavarian Alps. Tourists stranding during their trip with a BEV due to a depleted battery is tantamount to a severe setback in the acceptance of E-mobility technology. In the long run, insights into the structure of the reachability regions of BEVs will allow for a systematic and cost-efficient building of a network of charging stations.

For the purpose of a clean exposition we consider the following simplified model: A BEV is equipped with a battery of capacity M. Travelling along a road segment has one of the following two effects on the battery charge status: a) energy is consumed (typically the case when driving on a

[1] http://www.ee-tour.de

flat or uphill terrain) or b) energy is recuperated (when going downhill and/or braking). Two constraints have to be obeyed: i) at no point in time, the battery runs out of energy and ii) with the battery already at its full capacity M no energy recuperation takes place. For example, the *e-Smart*, a small 2-seat car,is equipped with a Lithium-Ion battery of capacity $M = 15.8kWh$, can recuperate energy during deceleration phases and has an average cruising range of about $110km$. To derive energy consumption/recuperation on a given road segment we combine the road network data from the OpenStreetMap project[2] with height information from the Shuttle Radar Topography Mission [3]. The result is a graph consisting of vertices and edges with the edge costs representing energy consumption (in case of positive edge costs) or recuperation (in case of negative edge costs).

In graph theory, a node v is *reachable* from a node u in a given (directed) graph $G(V, E)$, if there exists a (directed) path from u to v. The vertices u and v are called *connected* if either u is reachable from v or vice versa and *strongly connected* if both holds. In context of route-planning for BEVs we have to redefine the terms reachability and connectivity to account for the additional constraints i) and ii) on the battery charge mentioned above. A path from u to v in our graph does not imply that we can travel from u to v, as the battery charge status in node u might not allow for travelling along the path without running out of energy in between. Therefore in our context of BEV route planning we call v *ev-reachable* from u if there exists a path from u to v, that obeys the battery constraints. Similarly, we call v *strongly ev-connected* to u if there exists a round tour visiting v, that starts and ends in u and obeys the battery constraints. In Figure 1 gives an impression, how the sets of ev-reachable and strongly ev-connected nodes look like in practice. They are influenced by the height profile of the underlying terrain. As driving downhill can recuperate some energy, the BEV is able to get further in the middle of the map, where the streets follow a small valley. Note that for visualization purposes we have chosen an artificially small battery capacity in this and the other figures.

Related work

The problem of energy-optimal routing of BEVs in street networks was introduced in (Artmeier et al. 2010). There are two main differences to the conventional shortest path problem: On the one hand the edge costs might be partly negative, as BEVs are able to recuperate energy. On the other hand the battery constraints have to be fulfilled, namely a feasible path cannot contain a node with a battery load smaller than zero (running out of energy) or a battery load greater than the battery capacity M (overcharging). In (Eisner, Funke, and Storandt 2011) it was shown, that one-to-one-queries can be solved in time $\mathcal{O}(n \log n + m)$, by modelling the battery constraints as edge cost functions and using Johnson's shifting technique (Johnson 1977) to obtain non-negative edge costs. This approach is the basis for computing ev-reachable and ev-connected node sets efficiently.

[2] `http://www.openstreetmap.org`

[3] `http://www2.jpl.nasa.gov/srtm/`

Our Contribution

We extend the techniques developed in (Artmeier et al. 2010) and (Eisner, Funke, and Storandt 2011) to answer the above mentioned fundamental questions of BEV route planning. To that end, we show how to compute the set of ev-reachable nodes and how to determine the minimal battery load necessary to reach a given target. This requires deliberate modelling as edge cost functions to obtain efficient running times. Moreover we take battery switch stations into account and propose graph preprocessing techniques that allow the efficient computation of ev-reachable and ev-connected sets at query time even in the presence of BSSs. Furthermore we present an algorithm for energy-aware route planning minimizing the number of visited BSSs during a trip from A to B. Our algorithms are exact and efficient which is also reflected in our experimental evaluation.

EV-Reachable and EV-Connected Node Sets

We are given a street network $G(V, E)$ and a cost function $cost : E \to \mathbb{R}$ representing the energy consumption of the edges. We assume G to be free of any negative cycles as well as knowledge of the battery capacity M of the BEV. A path from s to t is called energy-optimal, if the final battery load in t is maximized by this path compared to all other paths. We can compute energy-optimal paths using a single run of Dijksta's algorithm (Dijkstra 1959), after an $\mathcal{O}(mn)$ preprocessing phase (as described in (Eisner, Funke, and Storandt 2011)). During this Dijkstra run we assign labels to the nodes representing the battery charge status. Initially these labels are $b(v) = -\infty \; \forall v \in V \setminus s$ and $b(s) = I$ with $0 \leq I \leq M$. Using a max-priority-queue (PQ) for the temporary labels, Dijkstra's algorithm then settles the nodes in *decreasing* order of battery charge status. The label of a settled node equals the maximal possible battery charge status we can reach this node with when starting in s.

EV-Reachable Node Sets

Computing the set of ev-reachable nodes requires to check for all nodes, if there exists a feasible path from the source node. Hence we can compute the set of ev-reachable nodes as $R(s) := \{v \in V |\; b_s(v) \geq 0\}$ with $b_s(v)$ being the final battery label, that was assigned to v by a Dijkstra computation starting in s with a fully charged battery ($b(s) = M$) and running until the PQ becomes empty. Note, that any node that gets pushed into the PQ during the Dijkstra run already has a feasible battery label at this point in time. Therefore the set of pushed nodes equals the set of ev-reachable nodes R and hence our algorithm has an output-sensitive runtime of $\mathcal{O}(|R| \log |R| + |E_R|)$ with E_R being the set of outgoing edges of the nodes in R. Clearly, this is $O(n \log n + m)$ but much smaller if the reachable nodes are only a small portion of the whole network (as usually the case).

Strongly EV-Connected Node Sets

The set of strongly ev-connected nodes is a subset of the ev-reachable node set, containing only the nodes, that also allow for returning to s without running out of energy. Note,

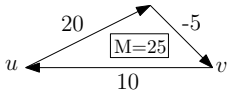

Figure 2: Strong ev-connectivity is not an equivalence relation: The roundtour is feasible when starting fully charged in u, but not feasible when starting in v (with a full battery charge of $M = 25$).

that in contrast to the conventional definition of strong connectivity, strong ev-connectivity is not an equivalence relation anymore as the reflexivity might be compromised due to the existence of edges with negative costs, see the example in Figure 2. To compute the strongly ev-connected nodes $C(s)$ for a source s, we could check for each node $v \in R(s)$ if a Dijkstra run from v with initial battery load $I = b(v)$ yields a feasible path back to s. The running time $\mathcal{O}(n^2 \log n + nm)$ of this approach is prohibitive, though.

To improve runtime, we do not want to decide individually for a given battery charge status on a node v if there exists a feasible path back to s, but instead compute for all nodes the minimal charge status $b_{min}(v)$, that is sufficient to complete the round tour. This leads to the following formal definition of the strongly ev-connected node set: $C(s) := \{v \in V |\ b_s(v) \geq b_{min}(v)\}$. Based on that we define a new edge cost function on the reverse graph, that allows for computing b_{min} for all nodes by starting a single computation in s. If the original edge $e = (v, w)$ has non-negative costs we have to add the costs of the edge to $b_{min}(w)$ to obtain the minimal necessary battery load in v. If the resulting value $b_{min}(v) = b_{min}(w) + cost(e)$ exceeds M, we cannot use this edge. Therefore we define the cost function c_e^+ as follows:

$$c_e^+(b_w) = \begin{cases} cost(e) & b_w \leq M - cost(e) \\ \infty & otherwise \end{cases}$$

Considering an edge $e = (v, w)$ with negative costs we subtract the absolute value of the costs from $b_{min}(w)$ to obtain $b_{min}(v)$, therefore $b_{min}(v) = b_{min}(w) + cost(e)$. This edge can always be traversed, but as $b_{min}(v)$ better not be negative, we have to set it to 0 if $b_{min}(w) + cost(e) < 0$. This can be modeled with the following definition of c_e^-:

$$c_e^-(b_w) = \begin{cases} -b_w & b_w < -cost(e) \\ cost(e) & otherwise \end{cases}$$

So different from the definition of c_e^+ we have to deal here with a partly negative edge cost function, prohibiting the application of Dijkstra's algorithm at this point. It is easy to see that the two cost functions can be expressed uniformly as follows with apropriate choice of l and u:

$$c_e(b_w) = \begin{cases} -b_w + l + cost(e) & b_w < l \\ cost(e) & b_w \in [l, u] \\ \infty & b_w > u \end{cases}$$

To apply Dijkstra or the Bellman-Ford algorithm (Bellman 1958), (Ford 1962) on a graph with edge cost functions in general, these functions have to fulfill the FIFO-property:

Definition 1 *A function $f : \mathbb{R} \to \mathbb{R}$ satisfies the FIFO (first-in-first-out) property, if $\forall x \leq y$ we have $x + f(x) \leq y + f(y)$.*

Lemma 1 *The cost function $c_e()$ fulfills the FIFO-property.*

Proof. Let $f(x) = c_e(x)$. If $x < l$ we have $x + f(x) = 0$, which is the smallest possible battery charge status and threfore the inequality is fulfilled. Otherwise, let y be larger than u. Then we get $y + f(y) = \infty$, which of course is greater or equal to any possible term on the left side. Otherwise $x, y \in [l, u]$, but then $f(x) = f(y) = cost(e)$ and hence the inequality is also fulfilled. ∎

The FIFO property allows employment of the Bellman-Ford algorithm, but in order to use Dijkstra we have to assure that the cost functions are non-negative. For constant edge costs and graphs without negative cycles, this can be achieved by Johnson's shifting technique (Johnson 1977). To that end a potential is assigned to every node and the new edge cost of $e = (v, w)$ equals the old edge cost plus the potential of v minus the potential of w. For appropriate choice of the potentials we can assure that all the transformed edge costs are non-negative and moreover the structure of the shortest path does not change. In (Eisner, Funke, and Storandt 2011) it was shown, that Johnson's shifting technique applies for FIFO edge cost functions, if the graph with modified edge costs $c'_e = \min_x c_e(x)$ does not contain negative cycles. This condition is fulfilled in our case, as $\min_x c_e(x) = cost(e)$ and the original graph with these constant edge costs was assumed to be free of negative cycles. The appropriate node potentials can then be derived by adding a dummy node to the original graph which is connected to all nodes of the graph. Using Bellman-Ford we compute in time $\mathcal{O}(mn)$ from the dummy node to all nodes in the graph shortest path distances which then also serve as node potentials. Note, that this is a preprocessing step, that only has to be performed once. Each subsequent query can then be answered in the already transformed graph, with non-negative edge cost functions.

At this point we can compute the minimal necessary battery charge status for every node v by running a single Dijkstra on the reverse graph starting in s with $b_{min}(s) = 0$ and $b_{min}(v) = \infty\ \forall v \in V \setminus s$ using a min-priority-queue. It remains then to check $\forall v \in R(s)$ if $b(v) \geq b_{min}(v)$, because the FIFO property assures that in this case a feasible path exists from v to s. So all in all we need for each query two Dijkstra computations with a runtime of $\mathcal{O}(n \log n + m)$ respectively and the verification procedure, which is linear in the number of nodes. Therefore the resulting runtime is $\mathcal{O}(n \log n + m)$. Looking more closely it turns out that in the second Dijkstra run on the reverse graph it makes no sense to push any node v with $b(v) = -\infty$, because they neither can be part of the strongly ev-connected nodes on their own nor influence $b_{min}(v)$ of any v that is element of $C(s)$. Hence our runtime depends again on the size of the set of ev-reachable nodes R and can be expressed as $\mathcal{O}(|R| \log |R| + |E_R|)$ with this time E_R being the whole set of adjacent edges to nodes in R.

Considering Battery Switch Stations

With an increasing number of BEVs on the streets the density of battery switch stations will grow as well. A battery switch station is a node $l \in V$, that leads to a battery load $b(l) = M$, whenever it is visited. The presence of BSSs can augment the set of reachable and connected nodes and might affect the most energy-efficient path. In this section we develop algorithms which take into account BSSs. We assume that the number of BSSs is not too large ($O(\sqrt{n})$), which seems a realistic assumption to make (for Germany which has about 14k gas stations this corresponds to several thousand BSSs).

Augmented Reachable Node Sets

We are given a source node $s \in V$ and additionally a set of BSSs $L \subseteq V$. We want to compute the sets of nodes, which are ev-reachable from s directly or over a feasible path, that visits one or more BSSs. The following approaches will work for any initial battery load in s, but as we want to compute the maximal ev-reachable set, we assume a fully loaded battery at the start and hence set again $b(s) = M$. We refer to the resulting set of nodes as $R^L(s)$.

Naively we can compute this augmented ev-reachable node set incrementally: At first we set $R_0 = \emptyset$ and $R_1 = R(s)$ and then incrementally $R_i = R_{i-1} \cup \bigcup_{l \in L \cap R_{i-1} \setminus R_{i-2}} R(l)$. We can stop as soon as $|R_i| = |R_{i-1}|$, which is then the desired result. If we have to compute $R(l)$ $\forall l \in R$ on demand we end up with a runtime of $\mathcal{O}(|L| n \log n + |L| m)$ for this naive approach. Of course the set $R(l)$ is invariant under the choice of s and therefore we could precompute it for all BSSs, taking time $\mathcal{O}(|L| n \log n + |L| m)$ and space $\mathcal{O}(|L| n)$. Subsequent queries need then a runtime of $\mathcal{O}(n \log n + m + |L| n)$, because computing $R(s)$ requires time $\mathcal{O}(n \log n + m)$ and as each node in the graph might be contained in $\mathcal{O}(|L|)$ reachable sets, we still need time $\mathcal{O}(|L| n)$ to compute their union.

The following method will also take $\mathcal{O}(|L| n \log n + |L| m)$ preprocessing time, but allows for a query time of $\mathcal{O}(n \log n + m)$, using only linear additional space.

We make use of an auxiliar graph $Q(L, F)$ which is constructed as follows: For every $l \in L$ we compute $R(l)$; there is an edge $(l, l') \in F$ if $l' \in R(l)$. Q obviously has space consumption $O(|L| + |F|) = O(n)$ under the assumption of a not too dense set of BSSs.

Answering a query starts again with computing $R(s)$ for the given source s conventionally, storing all contained BSSs along the way in a list. Afterwards we start a BFS in Q on the set of these BSSs in order to extract all indirectly ev-reachable BSSs. Having the complete set of ev-reachable BSSs $L' \subseteq L$, we can start a single Djkstra run on this set by setting the battery labels $b(l) = M$ $\forall l \in L'$ and $b(v) = -\infty$ $\forall v \in V \setminus L'$ and pushing all elements of L' into the PQ. The result of this Dijkstra computation is the assignment of a battery label to each node v, that denotes the highest possible charge status that we can get in v, starting at *any* BSS $l \in L'$. Therefore it only remains to build the union of the nodes, that were visited during this Dijkstra run, and $R(s)$ to receive the final set $R^L(s)$.

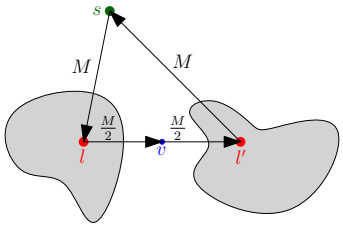

Figure 3: Node v is neither in $C(l)$ nor in $C(l')$ (gray areas), but belongs to $C^L(s)$ as the round tour s, l, v, l', s is feasible.

The runtime for a single query consists of two Dijkstra runs for computing $R(s)$ and $R(L')$, requiring time $\mathcal{O}(n \log n + m)$, a BFS computation with a runtime of $\mathcal{O}(m+n)$ and taking the union of two sets, requiring at most time $\mathcal{O}(n)$. Therefore the overall runtime is $\mathcal{O}(n \log n + m)$ and hence – different from the naive approaches – independent of the number of BSSs in the street network.

Augmented Connected Node Sets

Again we are given a source node s with $b(s) = M$ and a set of BSSs $L \subseteq V$. Now we want to determine all nodes $v \in V$, for which exists a feasible round tour from s to v and back with an arbitrary number of BSSs on the way.

Note, that different from computing $R^L(s)$ the augmented strongly ev-connected node set $C^L(s)$ is not equal to the union of the strongly ev-connected node sets of s and all directly or indirectly connected BSSs, but a superset as illustrated in Figure 3. Because of that the naive incremental approach now has to maintain two sets R and R' with R containing the ev-reachable nodes and R' the ones, that s can be reached from. So we have $R_0 = \emptyset, R_1 = R(s)$. The initialization of R' is $R'_0 = \emptyset$ and $R'_1 = R^{-1}(s) := \{v \in V | b_{s\ min}(v) < \infty\}$. The augmentation of the sets consists again of checking if in the last round new BSSs were added and if so take the union with $R(l)$ or $R^{-1}(l)$ for all such BSSs l. Moreover we have to remember for every node in R the maximal possible battery label and for R' the minimal necessary battery label. If both sets have their final size, we can check for all nodes in their intersection if the maximal battery label exceeds the minimal necessary battery charge. The set of nodes, for which this condition is fulfilled equals $C^L(s)$. Again the runtime and/or the space consumption scales badly with the number of BSSs in the network as the theoretical runtime is similar to the one of computing $R^L(s)$.

Fortunately our auxiliary graph $Q(L, F)$ can again help speed up the computation for this scenario as follows: For a given source s we compute $R(s)$ and $R^{-1}(s)$ conventionally, constructing $L_1 = L \cap R(s)$ and $L_2 = L \cap R^{-1}(s)$ along the way. Then we mark all BSS nodes in Q green, that are visited by a BFS run starting on L_1. Furthermore we mark all nodes red, that are visited in the reverse of Q starting on L_2. The set of nodes marked green *and* red L_{gr} equals then the set of strongly ev-connected BSSs for s. Knowing this total set, we can compute $R(L_{gr} \cup s)$ and $R^{-1}(L_{gr} \cup s)$ each with a single Dijkstra run in the graph with the respective edge cost functions, taking time

$\mathcal{O}(n \log n + m)$ as described in the previous subsection. Afterwards we also have to check for the nodes in $R(L_{gr} \cup s) \cap R^{-1}(L_{gr} \cup s)$, if the battery label $b()$ assigned during the computation of $R(L_{gr} \cup s)$ exceeds $b_{min}()$, that is determined by the Dijkstra run for $R^{-1}(L_{gr} \cup s)$. This needs only time linear in the size of the intersection, hence the total runtime remains $\mathcal{O}(n \log n + m)$. As a result we obtain the desired set of strongly ev-connected nodes $C^L(s)$.

One-to-one Queries

If we want to compute the optimal path from s to t with $s, t \in V$ in the presence of BSSs, the notion of 'optimal' has to be defined first. The path maximizing the final battery charge in t could be declared optimal. But this might lead to long detours over many BSSs and a high total energy consumption, though. So instead we aim for a feasible path requiring the minimal number of BSSs on the way.

Again our auxiliary graph $Q(V, F)$ (augmented with uniform edge costs) can be used to answer such a s-t-query. We first compute all the BSSs s can reach directly – let's call this set L_s – and all the BSSs t can be reached from – L_t – as before in time $\mathcal{O}(n \log n + m)$. Then we augment Q by a dummy source l_s and a dummy target l_t as well as zero-weight edges $(l_s, l) \forall l \in L_s$ and $(l, l_t) \forall l \in L_t$. In the augmented graph we start a Dijkstra from l_s to l_t. Let $l_s, l_1, v_2, \ldots v_{k-1} l_k, l_t$ be the resulting shortest path in the augmented graph. We output a feasible path from s to l_1 concatenated by the path from l_1 to l_k concatenated by a feasible path from l_k to t as final result. The two feasible paths from s to l_1 and l_k to t have been computed implicitly during the construction of L_s and L_t, respectively. It should be obvious that the approach returns the path with the minimal necessary number of recharging events and the overall running time is $O(n \log n + m)$.

Experimental Results

Our algorithms were evaluated on two benchmark graphs: The German Taunus with 11220 nodes and 24119 edges, and a map of Southern Germany with 5588146 nodes and 11711088 edges. We included the Taunus – a rather hilly region within Germany – as it contains relatively many edges of negative cost, one of the main challenges for an efficient algorithmic solution (Johnson's shifting technique has to be employed because of that). Our implementation were in C++, timings taken on a single core of an Intel i3-2310M processor with 2.1 GHz and 8 GB RAM.

In Table 1 we show our query times for computing reachable and connected node sets without considering BSSs. The maximal battery charge status M was chosen, such that the cruising range was about 125 km, matching the real cruising range of current BEVs. We can reach about 1/5 of the nodes in Southern Germany on average, when starting fully charged at a randomly chosen source. Our approach yields query times below a second for computing connected node sets. This is about factor 10^5 better than the naive way of checking for each node in $R(s)$ if there exists a feasible path back to s, although we sped them up using Contraction Hierarchies as described in (Eisner, Funke, and Storandt 2011).

	Taunus		Southern Germany	
	time(sec)	% nodes	time(sec)	% nodes
R(s)	0.00431	98.3	0.41872	19.0
C(s) naive	4.64245	98.3	96279.3	13.9
C(s)	0.00924	98.3	0.92853	13.9

Table 1: Query times for computing ev-reachable and strongly ev-connected node sets; percentage of covered nodes; averaged over 1000 random queries.

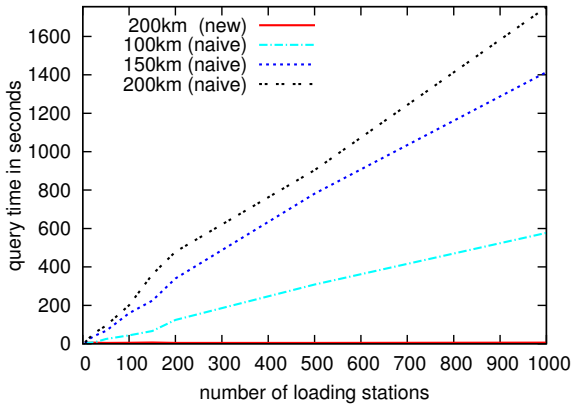

Figure 5: Query times for computing strongly ev-connected node sets in the presence of BSSs. Each plot line corresponds to a specific battery capacity, allowing to travel the given range on average.

Next we implemented the naive approaches as well as our new strategy for computing augmented sets of ev-reachable and strongly ev-connected nodes in the presence of battery switch stations. In Figure 4 all directly and indirectly ev-reachable nodes for a random selected source and twenty randomly placed BSSs are depicted. We can see in the lower part of the picture, that the ev-reachable BSSs enlarge the number of ev-reachable nodes only slightly, while in the upper part the number increases significantly, resulting in a third of all ev-reachable nodes being only reachable via BSSs. This is again due to the structure of the underlying terrain. If a BSS is close to the given source and the path from the source to the BSS has no uphill character, we arrive at the BSS almost fully charged anyway. The same yields for BSSs that are further away, but the path allows to recuperate most of the used energy. On the other hand, if a BSS can only be reached on a very energy-consuming path, we expect its ev-reachable nodes to be really different to the ones of the source, especially if the BSS allows to cross a peak.

The query times for the two approaches evaluated on the map of Southern Germany are shown in Figure 5. The timings are averaged over 10 rounds of randomly placing the given number of BSS in the map and 100 subsequent queries for each. The query times for the naive strategy – computing $C(l) \forall l \in L$ on demand – grows dramatically with the number of BSS as well as the maximal battery charge status and hence is only practicable for a very small number of BSSs and small cruising ranges. In contrast to that the runtime for the BFS based approach only grows slightly with

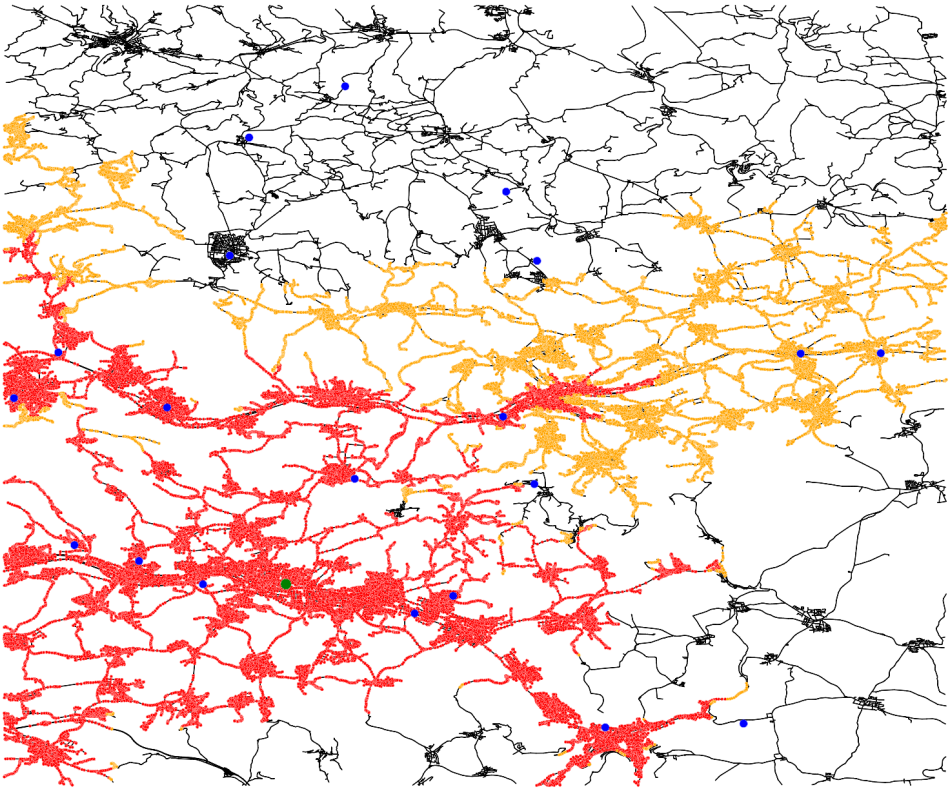

Figure 4: Augmented reachable node set for a source node (green). BBSs are marked blue. Red: nodes directly ev-reachabe; Orange: nodes indirectly ev-reachable.

the increasing battery capacity, namely from 1.14-4.06 for 100 km up to 4.46-6.21 seconds for 200 km.

Finally we measured the runtime of one-to-one queries making use of BSSs in the map of Southern Germany. The results can be found in Table 2. Again the BSSs were placed randomly in the street network. We computed the path, that required the minimal number of stops at battery switch stations. As expected the total query times are very similar to that for computing the set of reachable nodes, therefore we also achieve practicable runtimes below one second for answering one-to-one queries. In Figure 6 one can see two

cruising range	# BSS	no path (%)	direct (%)	indirect (%)	query time (sec)
80km	200	60.2	8.4	31.4	0.17
100km	100	38.0	19.4	42.6	0.32
100km	200	20.9	19.3	59.8	0.32
125km	50	10.8	35.6	53.6	0.47
125km	100	8,8	37.6	53.6	0.45
150km	50	8.0	52.8	39.2	0.53
150km	100	5.9	51.7	42.4	0.51

Table 2: Query times for one-to-one queries dependent on the number of BSSs and the possible cruising range. Moreover we recorded the percentages of paths, where the target is not ev-reachable from the source (no path), where the target is ev-reachable from the source without having to visit a BSS (direct) and where BSSs are necessary to receive a feasible path (indirect). All values are averaged over 1000 random queries.

examples of such paths. In both cases the target would not have been ev-reachable from the source without BSSs. In the upper image, only one visit of a BSS is necessary, in the lower one two recharging events occur. The paths are piecewise energy-optimal, hence avoiding going uphill as far as possible, following the valleys in the area. In both examples the paths reveal only small detours to visit the BSSs and hence these routes seem to be useful in practice.

Conclusions

In this paper we have extended the graph theoretic concepts of reachability and (strong) connectivity to the context of route planning for BEVs. The questions considered are of a very fundamental kind, still it seems as if we were the first to investigate them. In future work, we aim at incorporating advanced SpeedUp-techniques like (Bast et al. 2007) or (Geisberger et al. 2008) to improve the responsiveness of our query data structures. A more challenging question – maybe more relevant for the proliferation of E-mobility – is the development of optimization techniques for deployment of BSSs. If a budget for let's say 100 BSSs is available, how to determine the best locations for the new BSSs? Another direction of research is the development of more advanced objective functions for route planning. We have shown how to plan a route from A to B minimizing the stops at BSSs on the way. Such a route might take a long time, though, and there might be a much faster route which requires just one or two additional BSSs stops. A natural question could be: What is the most energy efficient feasible route that does not

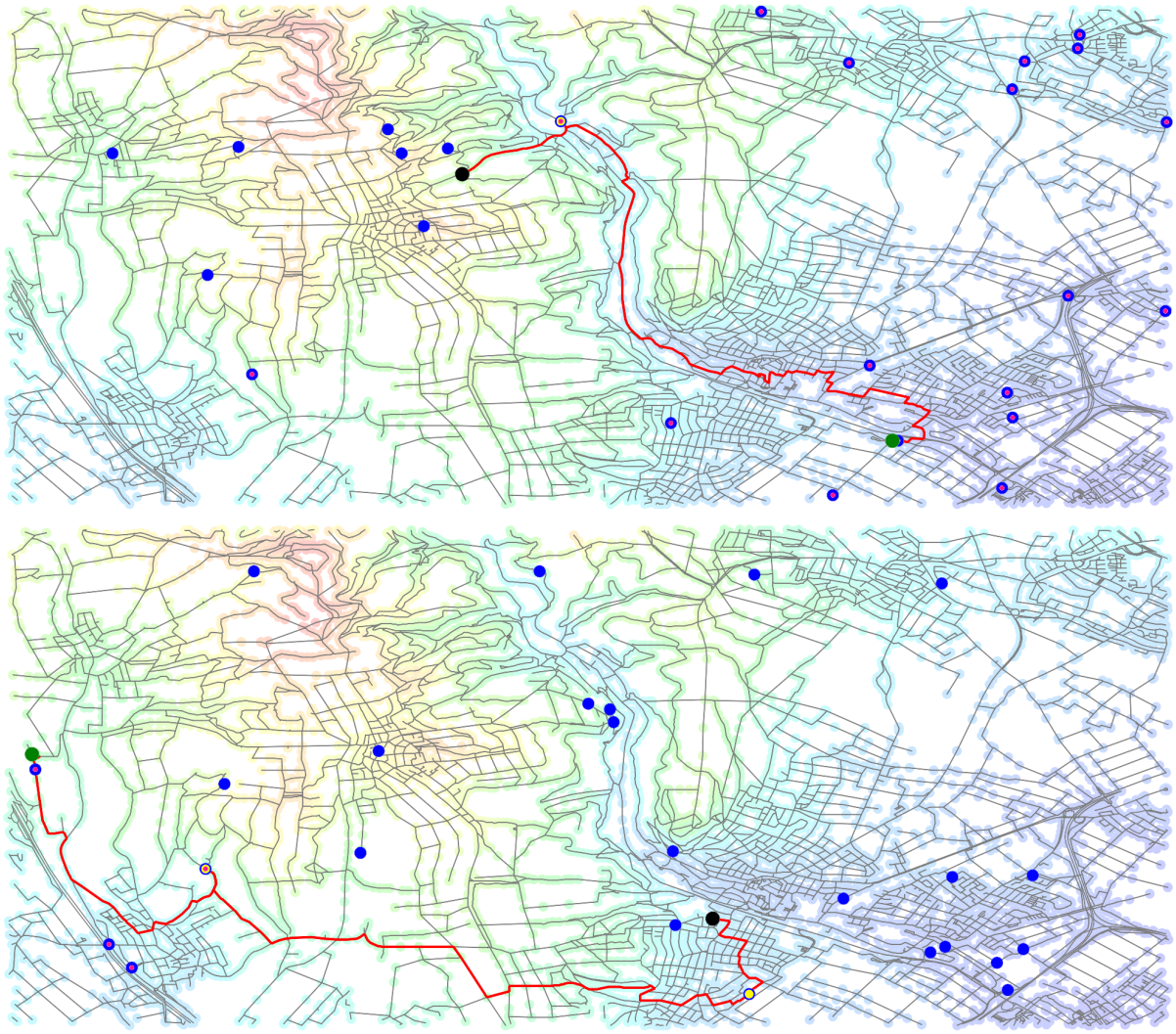

Figure 6: Example paths (red), where recharging is necessary to reach the target(black) from the source(green). BSSs are indicated by blue marks. All BSSs, that are ev-reachable from the source are marked pink and those actually selected for recharging are also coloured yellow. The desaturated node colours indicate elevation, ranging from 99 meters (deep blue) up to 412 meters (red).

take more than 10% longer than the fastest route? Unfortunately this turns out to be in instance of the NP-hard *constrained shortest path* problem, still, efficient approximation algorithms could be within reach.

References

Artmeier, A.; Haselmayr, J.; Leucker, M.; and Sachenbacher, M. 2010. The shortest path problem revisited: Optimal routing for electric vehicles. In *33rd Annual German Conference on Artificial Intelligence (KI)*.

Bast, H.; Funke, S.; Sanders, P.; and Schultes, D. 2007. Fast Routing in Road Networks with Transit Nodes. *Science* 316(5824):566.

Bellman, R. 1958. On a routing problem. *Quart. Appl. Math.* 16:87–90.

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

Eisner, J.; Funke, S.; and Storandt, S. 2011. Optimal route planning for electric vehicles in large networks. In *25th Conference on Artificial Intelligence (AAAI)*.

Ford, L. 1962. Flows in networks. *Princeton Univ. Press*.

Geisberger, R.; Sanders, P.; Schultes, D.; and Delling, D. 2008. Contraction hierarchies: faster and simpler hierarchical routing in road networks. In *Proc. of the 7th international conference on Experimental algorithms*, WEA'08, 319–333.

Johnson, D. B. 1977. Efficient algorithms for shortest paths in sparse networks. *J. ACM* 24:1–13.

Sachenbacher, M.; Leucker, M.; Artmeier, A.; and Haselmayr, J. 2011. Efficient energy-optimal routing for electric vehicles. In Burgard, W., and Roth, D., eds., *AAAI*. AAAI Press.