Planning Ahead for EV: Total Travel Time Optimization for Electric Vehicles

Sven Schoenberg and Falko Dressler

Abstract— **Travelling long distances with electric vehicles is becoming more viable today. Nevertheless, recharging is still necessary on long trips. As of now, the charging infrastructure is not yet ubiquitous and can be very heterogeneous in terms of charging power. Thus, appropriate route planning is needed, which is still an open research problem. We present an approach to optimize the total travel time for electric vehicles by selecting charging stations and routes, respectively, between origin and destinaton and the charging stations. We also take the possibility into account that driving below the speed limit helps to save energy. In particular, we use a multi-criterion shortest-path search to find the best compromise between the fastest and most economic route. In our approach, we use a non-linear charging model that supports CC-CV and CP-CV charging protocols used for lithium-ion batteries. To achieve acceptable speed for the multi-criterion shortest-path search, we combine contraction hierarchies with precomputation of shortest-path trees. By exploiting the fact that most routes are queried between the known locations of the charging stations, we were able to accelerate these queries by about two orders of magnitude. We compare our proposed adaptive charging and routing strategy to other strategies often cited in the literature. Our results clearly show that we are able to achieve a lower total travel time.**

I. Introduction

Battery capacities of electric vehicles are increasing and these cars can now drive much longer distances. Nevertheless, recharging on the trip is often still necessary. As of today, most charging infrastructure is deployed in bigger cities, with only few charging stations in the country side and on motorways. Due to the heterogeneous charging infrastructure in location and available charging power, the selection of the charging stations has a big impact on the total travel time of the trip. Thus, novel Intelligent Transportation Systems (ITS) are needed to help planning long distance trips ahead of time.

Route planning for electric vehicles is more challenging than traditional route planning, especially when taking recharging into account [1]–[3]. The time it takes to recharge depends on the consumed energy while driving and the power of the charging station. To optimize the total travel time, it might be better to drive a more economic route to save energy and, therefore, charging time, even if the pure driving time is longer. Instead of taking a different route, the driver may also save energy by driving below the speed limit, especially on roads with a high speed limit [4]. It might even make sense to take a detour to charge at a faster charging station. The amount of energy to charge at each charging station also influences the total travel time. Particularly, as charging stations do not charge with a constant power, but charge considerably slower after the battery has reached about 80 % [5].

Heinz Nixdorf Institute and Dept. of Computer Science, Paderborn University, Germany sven.schoenberg@c-lab.de, dressler@ccs-labs.org

The optimal route and driving speed depends on the power of the selected charging stations and the amount of energy charged at each station. Also, the selection of the charging stations and the amount of energy to charge depend on the selected route, so these aspects mutually influence each other. We approach this problem using a multi-criterion shortest-path search for the fastest and most economic routes. We then compare different route alternatives and select the best one. The problem is, that a multi-criterion shortest-path search is a lot more computationally expensive than a normal (single-criterion) shortest-path search [6].

In this paper, we present a way to accelerate these multi-criterion shortest-path searches by exploiting the fact that most queries are between the known locations of the charging stations. On that basis, we can create an adaptive strategy that optimizes charging station selection, routing and charging for optimal total travel time.

Out main contributions can be summarized as follows:

- We discuss models for energy consumption (III) and battery charging (IV);
- we present an approach to calculate trips for electric vehicles with minimal total travel time including charge stops (V); and
- we performed an experimental study to analyse the effect of different parameters on the preprocessing and query performance and compare our adaptive routing and charging strategy against other strategies (VI).

II. Related Work

The classic route planning is a shortest path problem, where the best route is to be found from A to B based on some criterion. Typically, the criterion to optimize for is either driving distance (shortest route), travel time (fastest route), energy consumption (economic route), or a combination thereof. The best known solution is the Dijkstra's algorithm [7]. Depending on the size of the graph, the algorithm can be too slow in practice, but several techniques have been proposed to speed things up. For example, the A* algorithm [8] uses a heuristic for a directed search. If the heuristic is guaranteed not to overestimate the cost, A* will find the optimal path.

Another technique are contraction hierarchies introduced by Geisberger et. al. [9]. In a preprocessing step, shortcuts are added to the graph that can later speed-up the path finding

query significantly. This is done by contracting the nodes of the graph one by one. Each node that is contracted, is effectively removed from the graph. If the node was part of the shortest path between two of its neighbors, a direct edge between these neighbors (shortcut) is added to ensure that the shortest path is maintained. Whether this is the case can be determined by doing a shortest path search with Dijkstra's algorithm from each neighbor to all other neighbors. Each node is assigned a level based on the order of contraction. A higher level indicates that the node was contracted later and its shortcuts might have replaced shortcuts of lower level nodes. To query the shortest path, a bidirectional search with Dijkstra's algorithm is done with both sides only traversing to nodes that have a higher level until they meet. This way, the number of nodes that need to be visited to find the shortest path is reduced significantly. To further speed-up the query, A* can be used instead of Dijkstra's algorithm for the bidirectional search [4].

Route planning for electric cars presents additional challenges. The constraints of the battery, especially the limited range, have to be accounted for. This can also include recuperation, i.e., charging the battery when braking or driving down a slope. Finding the shortest path that also considers the battery constraints is a Constrained Shortest Path (CSP) problem [10]. To find the fastest route that is reachable with a limited range, a multi-criterion shortest path search can be performed using the criteria travel time and energy consumption. This results in all Pareto optimal paths for these criteria and we can choose the one with the best travel time that still fits the energy constraint. To calculate all Pareto optimal paths, a modified version of Dijkstra's algorithm or A* can be used; however, due to the increased complexity, this is even more impractical than for a single criterion. Fortunately, contractions hierarchies can also be used to speed-up multi criterion path finding and to solve the CSP in acceptable time [11].

The preprocessing step for contractions hierarchies for multi-criterion path finding requires a lot of computational effort for large country-sized maps. As more and more nodes are contracted, the remaining uncontracted nodes get more and more neighbors, which makes contracting the last few nodes very expensive. It is possible to restrict the preprocessing to only contract a subset of all nodes. Storandt contracted only 99.5 % of the nodes to achieve reasonable preprocessing times [12]. The remaining uncontracted graph is called a core graph [13]. This can substantially save preprocessing time, but also causes higher query times.

Apart from the path itself the driving speed can be considered part of the optimization. Especially on freeways with very high speed limits or no speed limit at all, e.g., on the German Autobahn, it might make sense to drive well below that speed limit, or the maximum speed of the vehicle, to save energy. This complicates the preprocessing step of the contractions hierarchies, because the speed affects the drive time and energy consumption that are part of the decision of whether a shortcut is required or not. A trivial way to account for different driving speeds would be to preprocess each desired speed separately. This seems impractical because of the high amount of computational effort required for preprocessing and storing all the preprocessed data. Hartmann and Funke [14] presented a way to only preprocess once for a range of speeds and then being able to query it for any speed within that range.

Another approach by Baum et al. [4] introduces a way to solve the Electric Vehicle Constrained Shortest Path (EVCSP) problem, which they define as the path that respects battery constraints and minimizes overall travel time. It provides driving speed recommendations and does not need to calculate all Pareto optimal paths. They speed up the queries with a combination of contractions hierarchies and A* and can achieve optimal results below one second for realistic battery sizes within Europe. In many practical use cases, a fast query time is more important than having exact results. By using a heuristic, query times can be improved significantly at the cost of some inaccuracy [4], [14].

Another problem is selecting the charging stations, when recharging is necessary on a long trip. One solution is to limit the number of recharging events and choosing the most economic route that is not more than 10 % longer than the shortest route [11]. Most publications assume a full recharge at each charging station [6], [11], [15], only few also consider partial charging [1], [3].

We go one step further and consider partial charging using a realistic nonlinear charging model and also provide driving speed recommendations. Building upon contraction hierarchies, we introduce a way to significantly reduce the number of times we have to explore the graph when making queries to and from charging stations by using precomputed shortest-path trees.

III. Energy Model

For electric vehicle routing, the energy consumption is an important criterion. To estimate the energy consumption on the road, we use a simplified energy model. The most important factor in this model is the driving speed. In general, the energy consumption increases with speed due to increased friction and air drag. Speed independent energy consumption results from, e.g., lights, radio, and air conditioning. Additionally, at lower speeds, we assume that there is more energy consumption due to acceleration and deceleration in areas with many traffic lights and higher traffic density. We assume that there is an optimal speed of $v_{opt} = 65\,\mathrm{km/h}$. At this speed, the vehicle has a base energy consumption $B_{base} = 0.16\,\mathrm{kWh/km}$. If we drive slower than that, the energy consumption will increase. The optimal speed has been chosen to be between inner city speeds which are typically up to 50–60 km/h and speeds on rural roads and freeways which are typically more than 70 km/h. The base energy consumption was chosen as a generic energy consumption of a typical electric vehicle.

We use the following simplified energy consumption model to calculate the energy consumption:

$$B = B_{base} + \frac{(v - v_{opt})^2}{100000} \ . \tag{1}$$

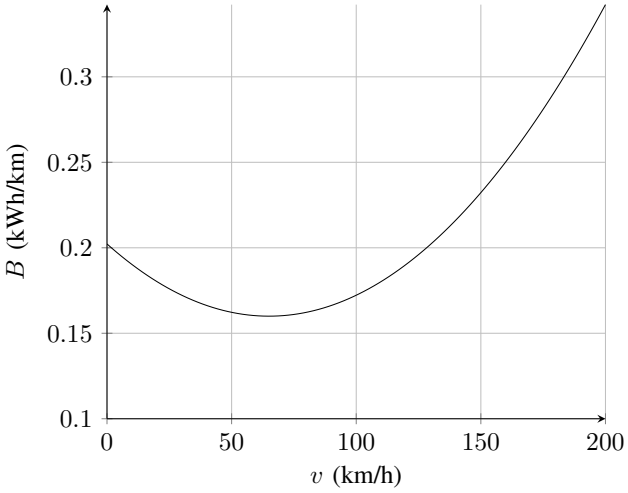


Fig. 1. Energy consumption B depending on driving speeds v

The model is plotted in Figure 1. It neglects some aspects like differences in elevation or traffic density, however, it is a sufficient to evaluate our approach.

IV. Charging Model

Many publications assume charging speed to be constant [1], [3]. In reality, however, the charging speed of the typical lithium-ion batteries decreases considerably after the battery's State of Charge (SOC) has reached about 80 % [5].

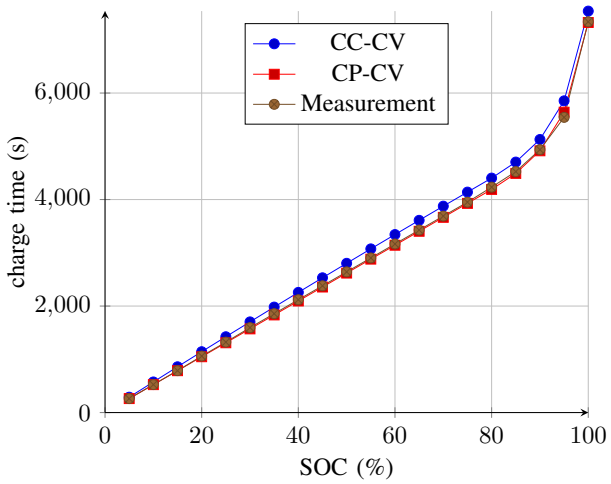
Lithium-ion batteries are typically charged with the CC-CV (constant current - constant voltage) charging protocol [5], [16], [17]. The charging consists of two phases, at first the battery is charged with a constant current while the charge voltage rises. When the charge voltage reaches 4.2 V, the second phase begins and the charge voltage is kept constant to prevent overcharging. At this point the battery has an SOC of about 80 %. The charge current then steadily decreases and when it falls below a threshold the charging is complete. An alternative protocol is CP-CV (constant power - constant voltage), where the power is kept constant in the first phase instead of the current.

Our charging model supports both the CC-CV and the CP-CV charging protocols. For simplification, we assume that the phases switch exactly at 80 % and that the voltage increase and current decrease is linear to the SOC [5]. In our model, p_{max} is the maximum charging power of the charging station. soc is the SOC of the battery, and can be between 0 and 1. At a state of charge of $soc = 0\,\%$, the voltage is defined as $u_{low} = 3.8\,\mathrm{V}$. The maximum voltage for $soc = 80$–$100\,\%$ is $u_{high} = 4.2\,\mathrm{V}$. The maximum current is calculated as

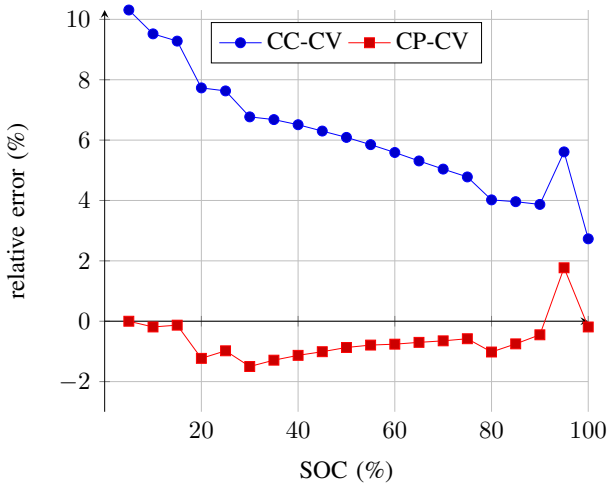
$$i_{max} = \frac{p_{max}}{u_{high}} \,. \tag{2}$$

For the CC-CV charging protocol, we assume a constant current and rising voltage until 80 % and after that a constant voltage and declining current linear to the SOC:

$$i(soc) = \begin{cases} i_{max} & \text{for } soc < 0.8 \\ \frac{1-soc}{0.2} \cdot i_{max} & \text{for } soc \geq 0.8 \end{cases} , \tag{3}$$



(a) Absolute charge time



(b) Relative error compared to measurement data

Fig. 2. Comparison of CC-CV and CP-CV charging protocols with measurement data.

$$u(soc) = \begin{cases} u_{low} + \frac{soc}{0.8}(u_{high} - u_{low}) & \text{for } soc < 0.8 \\ u_{high} & \text{for } soc \geq 0.8 \end{cases} , \tag{4}$$

$$p_{cc\text{-}cv}(soc) = u(soc) \cdot i(soc) \,. \tag{5}$$

For the CP-CV charging protocol, we can assume constant power until $soc = 80\,\%$ as

$$p_{cp\text{-}cv}(soc) = \begin{cases} p_{max} & \text{for } soc < 0.8 \\ u(soc) \cdot i(soc) & \text{for } soc \geq 0.8 \end{cases} . \tag{6}$$

The power is recalculated after each second of charging and terminates when $soc = 99\,\%$. We compared our model to measurement data from charging an electric vehicle [18]. The source of the data did not specify the charging protocol, but as can be seen in Figure 2 with CP-CV, our model matches the measured charging time within ±2 %, while with the CC-CV protocol it has a relative error of more than 10 % at the beginning. Therefore, the vehicle was apparently charged with the CP-CV charging protocol.

V. Charging Station Routing

In our charging station routing algorithm, we try to find a route for an electric vehicle by minimizing the total trip time including stops at charging stations. We define the total trip time as the sum of drive time and charge time. The charge time at the charging stations depends on the charge power and the amount of energy that needs to be charged. Therefore, it might be faster overall to choose an economical route over a fast route, if it saves time at the charging station. Apart from the route itself, the driving speed also has a huge influence on the time and energy consumption. Especially on freeways, e.g., the German Autobahn with very high speed limits or no speed limit at all, it might make sense to drive below the speed limit, or the maximum speed of the vehicle, to save energy.

Taking all this into account, our approach consists of two parts. The first part is a multi-criterion shortest path search, that is also able to query routes with different maximum driving speeds. The result is a set of Pareto-optimal routes from fastest to most energy efficient with a maximum driving speed suggestion for each.

The second part is the routing and charging strategy that uses the multi-criterion shortest path search to find routes between the charging stations and the origin and destination. With this, it can select the optimal charging stations for recharging and the routes between them to minimize the total travel time.

A. Multi-Criterion Shortest Path Finding

Multi-criterion shortest path finding can be done with a modified version of Dijkstra's algorithm. To accelerate query times, we use contractions hierarchies. The contractions hierarchies are generated for a maximum driving speed and a query will only result in the Pareto set of routes with this maximum speed. To also generate results for another maximum speed, we can recalculate the edge weights including shortcuts for the new maximum speed before the query. This only works for maximum speeds below the maximum speed that the contraction hierarchy was generated for. We assume that higher maximum speeds will mostly lead to additional routes and, therefore, to additional shortcuts in the contraction hierarchy compared to lower maximum speeds. This is not always the case as we will discuss in Section VI. To get a combined result from a Pareto set of routes together with speed suggestions, we have to iteratively query different maximum speeds and combine them into one Pareto set.

The contraction hierarchies greatly improve the query times, but for long distances of $>$200 km, on a complex graph the query times might still be in the order of seconds or even minutes. Considering that we might have to make many queries between charging stations, this could lead to unacceptable overall computation times.

B. Shortest-Path Tree Precomputing

Much of the computation time of queries is spend on exploring the graph and creating Pareto sets of labels at each visited node. We can take advantage of the fact that all queries are between the charging stations and the origin and destination. In an additional preprocessing step, we explore the graph from each charging station and thereby create shortest-path trees. We can limit the exploration to an energy consumption equal to the battery capacity of the vehicle. Because we use contraction hierarchies, the number of explored nodes is not excessively high and we can precompute the shortest-path trees for all charging stations. Due to the fact that the street network is a directed graph, we have to precompute two shortest-path trees for each charging station: One exploring the graph forwards, and one exploring backwards.

For each query, we take the forward shortest-path tree of the origin node and the backward shortest-path tree of the destination node. We find the nodes that are covered by both trees, i.e., nodes that have Pareto sets of labels in both trees. For each of these nodes, we create the sumset of both Pareto sets and remove all non-Pareto optimal elements. The result is the set of costs of the Pareto optimal shortest-paths from origin to destination that contain this node. By creating the Pareto set of all these sumsets, we get the set of costs of all Pareto optimal shortest-paths.

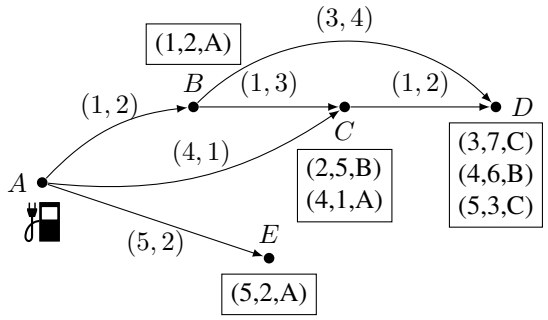
An example of such a query with shortest-path trees for two criteria is depicted in Figure 3. To be able to reconstruct the actual paths, each label contains the predecessor node additionally to the cost. When creating the sumsets for the common nodes, the predecessor nodes from both labels have to be stored. The combined Pareto set of all sumsets then also stores the node of the sumset. Each element of the resulting set then contains sufficient information to reconstruct its path.

For the queries in our charging station routing algorithm, we first have to create the shortest-path trees for the origin and destination. All queries can then be done without having to explore the graph again. To save time, we do not have to reconstruct all paths; we can select a path based on the costs and only reconstruct the selected one.

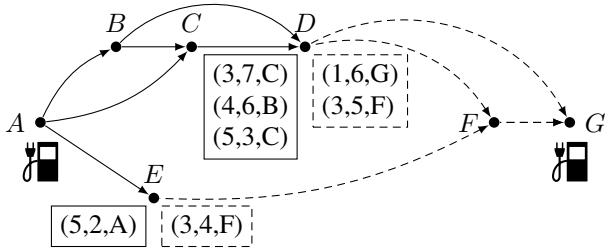
C. Routing and Charging Strategy

Our approach aims at selecting the charging stations and finding a route between the charging stations as well as a suggested maximum driving speed V_{max} that minimizes the total travel time. To select the charging stations, we create a graph that connects the start, the destination, and all charging stations. On this graph, we query the shortest path from start to destination with an A* search. To calculate the edge weights of the graph, we query the Pareto optimal set of routes with the multi-criterion shortest path search and select the route that has the smallest combined charge and drive time.

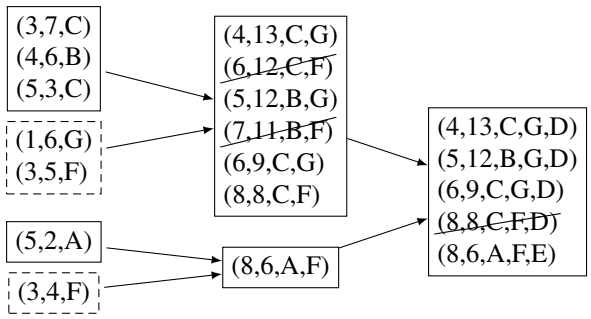
The charge time depends on the charge amount and the power of the charging station according to the charging model (cf. Section IV). Many publications assume a full charge at every charge stop. We propose an adaptive charging strategy where the charge amount depends on the charge power of the current charging station and the following one. We, of course, always charge at least the energy required to be able



(a) Shortest-path tree with pareto sets of labels



(b) Forward and backward shortest-path trees with labels of common nodes. Solid: Forward tree from node A. Dashed: Backward tree from node G



(c) Create sumsets from node labels and combining them to get resulting Pareto set

Fig. 3. Example query with precomputed shortest-path trees

to reach the next charging station. If the current charging station has a higher charge power than the following one, we continue to charge until the charging power drops below the maximum charge power of the next charging station.

When selecting the route from the Pareto set that minimizes the charge time and drive time, we also take the current charge time and the charge time at the following charging station into account. Because we cannot know the charge amount at the following charging station yet, we simply assume a full charge. By including the charge time at the following station, we make sure that the route is selected based on how much time it will take to charge the energy required for this route. We propagate the SOC of the vehicles battery while exploring the graph.

As the calculation of all Pareto optimal routes is computationally expensive, even with shortest-path tree precomputing, we have to minimize the number of queries. The calculation of single-criteria shortest paths is multple orders of magnitude faster. Thus, when we explore a node, we set the edge weight to its neighbors only to an estimated value using the fastest and most energy efficient cost that are calculated by two single-criteria shortest path searches as a heuristic. Before the edge is actually travelled, the heuristic edge weights are replaced with the accurate values calculated with a multi-criteria shortest path search. When replacing the edge weight, the edge may no longer be at the top of the open list. In this case, we have to repeat the process until the edge at the top of the open list has an accurate (non-heuristic) edge weight. To further reduce the number of unnecessarily explored nodes, instead of using the linear distance as a heuristic value for A*, we query the single criteria shortest path.

VI. Experimental Results

A. Experimental Setup

All experiments were run on a quad-core Intel Xeon E3-1275 CPU at 3.4 GHz and 32 GB of memory. We implemented our algorithm in C and compiled with GCC 7.3.0 with the highest optimization setting (-O3).

For our experiments, we extracted a road network from OpenStreetMap.[1] Because the preprocessing of contraction hierarchies for multi criterion routing requires a lot of computational effort, we ran most experiments on the road network of the German state North-Rhine Westphalia (NRW) with a total of 4,417,600 nodes. Many of these nodes only have decorative purposes to model the shape of the road, only 786,935 nodes have more than two edges. Some experiments were also conducted on the road network of entire Germany (DE) with 26,130,892 nodes of which 4,295,175 have more than two edges.

B. Core Graph

Contracting all nodes of a large country-sized map to generate contraction hierarchies for multi criterion routing can be quite time consuming. To achieve reasonable preprocessing time, we can stop contracting nodes after a certain amount has been reached, leaving an uncontracted core graph. This can save preprocessing time, but also causes higher query times.

We first evaluated how the amount of contracted nodes affects the preprocessing and query times. Thus, we generated 400 random origin-destination (OD) pairs with distances of 50, 100, 150 and 200 km. As can be seen in Figure 4, contracting all nodes may take several hours, while contracting 99.90 % takes less than five minutes. However, the average query time is then several times higher. Contracting 99.97 % of the nodes seems to be a good compromise for practical purposes. Preprocessing to 99.98 % takes significantly more time with little benefit for the average query time.

C. Variable Speed

The preprocessing time also depends on the maximum driving speed V_{max}, which the contraction hierarchy is build for. A higher V_{max} makes the preprocessing more expensive,

[1] Downloaded from download.geofabrik.de on 2018-04-23. All OSM ways with "highway" tag except for path, steps, elevator, corridor, platform, bridleway, footway, cycleway, pedestrian, proposed, construction, raceway, emergency_bay, rest_area or track.

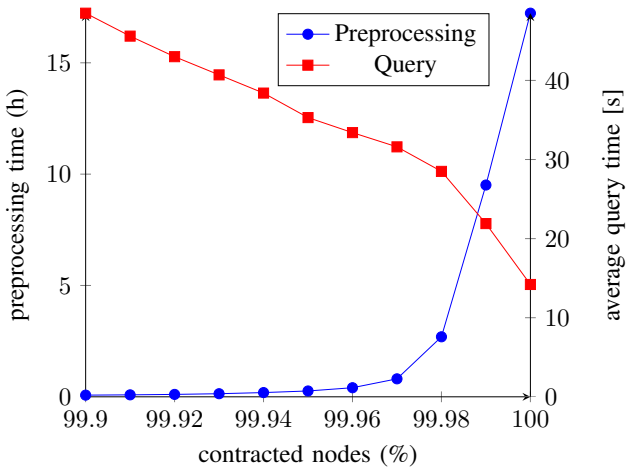


Fig. 4. Effect of amount of contracted nodes on preprocessing time and average query time

TABLE I
PREPROCESSING TIMES FOR DIFFERENT V_{max} VALUES

V_{max}	Visited nodes	Preprocessing time	Shortcuts
90	18,347,029,512	01:28:00	6,271,106
110	59,798,528,416	06:31:24	6,385,736
130	129,126,913,941	16:36:56	6,532,007

as it will usually increase the number of possible Pareto paths and therefore the number of shortcuts. In Table I, we compare the preprocessing times for different V_{max} values. All times are for 100 % contracted nodes.

It can be seen that it is infeasible to preprocess contraction hierarchies for every possible V_{max} value. As outlined in Section V-A, we can query routes using a contraction hierarchy that was preprocessed for a higher V_{max}. We evaluated how this affects the quality of the results by comparing the routes queried from a contraction hierarchy with a higher V_{max} with known correct results. In Table II, we show that we got correct results for at least about 80 % of all queries. When comparing the Pareto routes of the incorrect results with the known correct results, we found that at least 95 % of the routes were correct.

D. Heuristics

Contraction Hierarchies can offer a substantial performance improvement, but the query times can still be too high for some practical use cases. The query time correlates to the number of Pareto optimal routes. For the distance of about

TABLE II
RESULT QUALITY WHEN CALCULATING RESULTS WITH A CONTRACTION HIERARCHY (CH) WITH DIFFERENT V_{max} VALUE

CH V_{max}	Query V_{max}	Exact result	Correct routes
130	110	86.25	98.74
130	90	79.25	95.68
110	90	80.50	97.47

TABLE III
AVERAGE NUMBER OF ROUTES AND QUERY TIMES FOR DIFFERENT HEURISTIC VALUES AND OD-PAIR DISTANCES

ε	50 km		100 km		150 km		200 km	
0.0	130	2.9 s	286	10.7 s	708	34.4 s	1157	80.4 s
0.1	43	1.4 s	54	4.6 s	66	10.2 s	67	16.7 s
0.5	19	0.8 s	21	2.4 s	23	4.8 s	21	7.3 s
1.0	13	0.6 s	15	1.7 s	15	3.3 s	14	4.8 s
5.0	6	0.3 s	7	0.7 s	7	1.2 s	6	1.5 s
10.0	4	0.2 s	5	0.5 s	5	0.8 s	4	0.9 s

200 km in NRW, most queries in our random test set result in more than 1000 routes, many of which are very similar to each other. In practical use cases, knowing all optimal routes is often not necessary. We can eliminate very similar routes by setting up a constraint that each route must at least have an improvement of ε over the other routes.

In Table III, we can see that the value of ε has a big influence on the number of routes and the query time. A value of $\varepsilon = 0.1\,\%$ means that each route must have at least 0.1 % less energy consumption than the other routes in the Pareto set.

E. Shortest-Path Tree Precomputing

As discussed, by precomputing the shortest-path trees of all charging stations, we can potentially save a lot of query time in our charging station routing algorithm. The shortest-path trees exploration is limited to an energy consumption based on the battery size. The battery size therefore has a big influence in the precomputing time and required disk space for all shortest-path trees. As can be seen from our results presented in Table IV for the NRW map (781 charging stations), the precomputation takes 30 min for a 20 kWh battery and results in a size of 2.5 GB. Using a 40 kWh battery, the process takes more than 10 h and nearly triples the size. The step from 40 kWh to 60 kWh batteries is less dramatic, because the NRW map is so small, that it limits the exploration by the size of the map itself. The map of Germany (4552 charging stations) is bigger and is therefore not limiting the exploration as much. The precomputation for a 40 kWh battery took 5 d and 200 GB of disk space. We did not precompute for a 60 kWh battery, because it takes an impractical amount of time and disk space.

Using a heuristic of $\varepsilon = 1.0\,\%$, we could reduce the precomputing time for 40 kWh batteries to about 3 h and 9 GB of disk space. We were also able to precompute for 60 kWh batteries in 6.5 h with 18 GB of disk space.

To compare the query times between precomputed shortest-path trees and plain contraction hierarchies, we queried routes for OD-pairs with different distances, where both origin and destination are charging stations. We used precomputed shortest-path trees for 60 kWh batteries, with and without a heuristic value of $\varepsilon = 1.0\,\%$. For the map of Germany without heuristics, we used 40 kWh, because we could not precompute for larger values. For that reason, we also did not query 400 km distances, because they are typically not doable with a 40 kWh battery.

TABLE IV

SHORTEST-PATH TREE PRECOMPUTING TIMES AND SIZES

Map	Battery	ε	Time	Size	Size per CS
NRW	20 kWh	0.0 %	00:30:39	2.5 GB	3.2 MB
	20 kWh	1.0 %	00:20:08	0.3 GB	0.4 MB
	40 kWh	0.0 %	10:11:11	7.0 GB	8.8 MB
	40 kWh	1.0 %	00:30:46	0.5 GB	0.7 MB
	60 kWh	0.0 %	11:47:51	7.9 GB	9.8 MB
	60 kWh	1.0 %	00:31:53	0.6 GB	0.7 MB
DE	20 kWh	0.0 %	08:09:18	20.0 GB	4.0 MB
	20 kWh	1.0 %	00:45:07	2.2 GB	0.5 MB
	40 kWh	0.0 %	116:57:34	200.0 GB	45.0 MB
	40 kWh	1.0 %	03:07:42	8.7 GB	1.9 MB
	60 kWh	0.0 %		not feasible	
	60 kWh	1.0 %	06:30:59	17.9 GB	4.0 MB

TABLE V

QUERY TIMES OF PRECOMPUTED SHORTEST-PATH TREES COMPARED WITH PLAIN CONTRACTION HIERARCHIES (CH); R: RECONSTRUCTING ALL PATHS, H: PRECOMPUTATION WITH A HEURISTIC

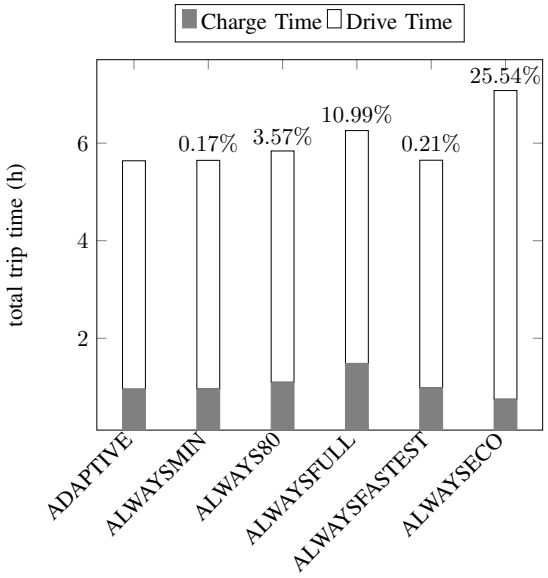
Map	Type	Average query times			
		50 km	100 km	150 km	200 km
NRW	Plain CH	2.861 s	10.716 s	34.420 s	80.354 s
	SPT	0.051 s	0.072 s	0.086 s	0.114 s
	SPT (R)	0.098 s	0.225 s	0.536 s	1.352 s
	SPT (H)	0.007 s	0.007 s	0.006 s	0.005 s
	SPT (R, H)	0.015 s	0.025 s	0.037 s	0.056 s
		100 km	200 km	300 km	400 km
DE	Plain CH	6.097 s	60.671 s	290.586 s	961.239 s
	SPT	0.450 s	0.492 s	0.693 s	-
	SPT (R)	0.486 s	1.665 s	4.286 s	-
	SPT (H)	0.037 s	0.039 s	0.033 s	0.030 s
	SPT (R, H)	0.054 s	0.125 s	0.219 s	0.291 s

Reconstructing all paths of the resulting Pareto set can take a significant amount of time and is not necessary for our algorithm. To compare the query time difference, we have tested it with and without reconstructing all paths. As can be seen in Table V, the query times for precomputed shortest-path trees are about two orders of magnitude smaller than for plain contraction hierarchies. Reconstructing all paths increases the query time especially for long distances. Using precomputed shortest-path trees with a heuristic, the query time is an additional order of magnitude smaller, at the cost of some accuracy.

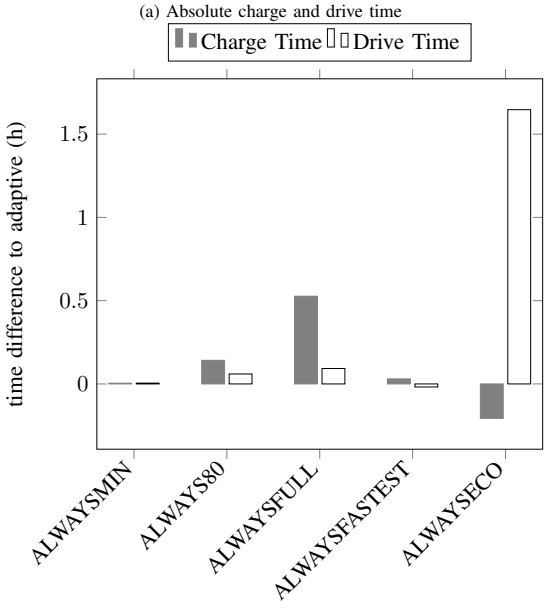
F. Charging Station Routing

For all experiments involving charging station routing, we use the whole map of Germany. We have generated a contraction hierarchy with 99.97 % contracted nodes. The preprocessing took 8 h and 16 min. We generated 30 random OD-pairs with distances of 300, 400 and 500 km. We consider all charging stations from the list provided by German Bundesnetzagentur.[2] It contains 4638 charging stations all over Germany with charging powers ranging from 3.7–350 kW. For the tests, we assume an electric vehicle with a 40 kWh battery.

[2]https://www.bundesnetzagentur.de/DE/Sachgebiete/ElektrizitaetundGas/Unternehmen_Institutionen/HandelundVertrieb/Ladesaeulenkarte/Ladesaeulenkarte_node.html (visited on 07/09/2018).



(a) Absolute charge and drive time



(b) Difference to adapative strategy

Fig. 5. Comparison of adaptive charging and routing strategy to other strategies.

We compared our adaptive routing and charging strategy (cf. Section V-C) to strategies often found in the literature. We further compared the charging strategy to always completely charging the battery at every stop, always charging to 80 % and charging just enough to get to the next stop. The routing strategies compared were always choosing the most energy efficient route and always choosing the fastest route that is still reachable with the battery capacity.

The results are plotted in Figure 5. As can be seen, our adaptive charging and routing strategy outperforms all other strategies. The charging strategy to always make a full charge at each stop (ALWAYSFULL), is about 11 % slower. It leads

to a lot more charge time because the charge power drops significantly after 80 % and it is probably charged more than necessary to get to the destination. The drive time also increases slightly, because it is now more beneficial for the routing algorithm to drive detours to faster charging stations. The same effect can be seen for the strategy to always charge to 80 % (ALWAYS80), albeit to a lesser extent.

The difference to only charging the minimum amount to get to the next charging station (ALWAYSMIN), is negligibly small. However, our adaptive algorithm is advantageous by utilizing charge power differences, i.e., charging more at the first fast charging station to save time at the second slower charging station. Apparently, in the used setting, these differences are rather small as many free fast charging stations are available.

The routing strategy to always choose the most economic route (ALWAYSECO) has the biggest disadvantage of about 26 % more total travel time compared to our adaptive routing strategy. Economic routes often take slower roads and, therefore, have a significantly longer drive times. Even though a lot of energy and, therefore, charge time could be saved, this does not make-up the lost time on the road. Always choosing the fastest route (ALWAYSFASTEST) is not much worse compared to our adaptive routing strategy. Apparently, enough free fast charging stations exist, to make this a valid approach. Again, when current availability is considered, this would change substantially.

VII. Conclusion

We presented an approach to optimize the total travel time for electric vehicles by selecting charging stations and the routes between the charging stations and the origin and destination. The greatest challenge was to accelerate the multi-criterion shortest-path search to be usable on large country-sized maps. We achieved this by combining contraction hierarchies with precomputing shortest-path trees. By exploiting the fact that most routes are queried between the known locations of the charging stations, we were able to accelerate these queries by about two orders of magnitude. For country sized maps, the query times are still too high for interactive applications. We compared our adaptive charging and routing strategies to several others often found in literature. Our results clearly show that our adaptive charging strategy has many advantages compared to the other solutions.

Future work includes developing concepts to improve the query times for charging station routing. We also want to take waiting times at the charging stations into account. Aside from selecting charging stations with less expected waiting time, we could select a slower, more energy efficient route when we know that we have to wait at the charging station anyway, or take a faster route if it means arriving before others, comparing selfish and cooperative strategies.

Acknowledgement

This work was partially supported by the German Federal Ministry of Education and Research (BMBF) through the APPSTACLE project (01IS16047E).

References

[1] M. Alizadeh, H.-T. Wai, A. Scaglione, A. Goldsmith, Y. Fan, and T. Javidi, "Optimized Path Planning for Electric Vehicle Routing and Charging," in *52nd Annual Allerton Conference on Communication, Control, and Computing (Allerton 2014)*, Monticello, IL: IEEE, Sep. 2014, pp. 25–32.

[2] C. Liu, J. Wu, and C. Long, "Joint Charging and Routing Optimization for Electric Vehicle Navigation Systems," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 2106–2111, Aug. 2014.

[3] M. Baum, J. Sauer, D. Wagner, and T. Zündorf, "Consumption Profiles in Route Planning for Electric Vehicles: Theory and Applications," in *16th International Symposium on Experimental Algorithms (SEA 2017)*, London, UK: Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Jun. 2017, 19:1–19:18.

[4] M. Baum, J. Dibbelt, D. Wagner, and T. Zündorf, "Modeling and Engineering Constrained Shortest Paths for Battery Electric Vehicles," in *25th Annual European Symposium on Algorithms (ESA 2017)*, Vienna, Austria: Schloss Dagstuhl-Leibniz-Zentrum für Informatik, Sep. 2017, 11:1–11:16.

[5] K. S. Ng, C.-S. Moo, Y.-P. Chen, and Y.-C. Hsieh, "Enhanced coulomb counting method for estimating state-of-charge and state-of-health of lithium-ion batteries," *Applied Energy*, vol. 86, no. 9, pp. 1506–1511, Sep. 2009.

[6] M. T. Goodrich and P. Pszona, "Two-Phase Bicriterion Search for Finding Fast and Efficient Electric Vehicle Routes," in *22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL 2014)*, Dallas, TX: ACM, Nov. 2014, pp. 193–202.

[7] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959.

[8] P. E. Hart, N. J. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, Apr. 1968.

[9] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," in *Experimental Algorithms 7th International Workshop (WEA 2008)*, Provincetown, MA: Springer, May 2008, pp. 319–333.

[10] A. Artmeier, J. Haselmayr, M. Leucker, and M. Sachenbacher, "The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles," in *KI 2010: Advances in Artificial Intelligence (KI 2010)*, Karlsruhe, Germany: Springer, Sep. 2010, pp. 309–316.

[11] S. Storandt, "Quick and Energy-Efficient Routes - Computing Constrained Shortest Paths for Electric Vehicles," in *5th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS 2012)*, Redondo Beach, CA: ACM, Nov. 2012, pp. 20–25.

[12] ——, "Route Planning for Bicycles - Exact Constrained Shortest Paths Made Practical Via Contraction Hierarchy," in *22nd International Conference on Automated Planning and Scheduling (ICAPS 2012)*, Atibaia, Brazil: AAAI Press, Jun. 2012, pp. 234–242.

[13] J. Dibbelt, T. Pajor, and D. Wagner, "User-Constrained Multi-Modal Route Planning," in *Meeting on Algorithm Engineering & Experiments (ALENEX 2012)*, Kyoto, Japan: Society for Industrial and Applied Mathematics, Jan. 2012, pp. 118–129.

[14] F. Hartmann and S. Funke, "Energy-Efficient Routing: Taking Speed into Account," in *KI 2014: Advances in Artificial Intelligence (KI 2014)*, Stuttgart, Germany: Springer, Sep. 2014, pp. 86–97.

[15] Z. Sun and X. Zhou, "To save money or to save time: Intelligent routing design for plug-in hybrid electric vehicle," *Transportation Research Part D: Transport and Environment*, vol. 43, pp. 238–250, Mar. 2016.

[16] M. Etezadi-Amoli, K. Choma, and J. Stefani, "Rapid-Charge Electric-Vehicle Stations," *IEEE Transactions on Power Delivery*, vol. 25, no. 3, pp. 1883–1887, Jul. 2010.

[17] S. Bashash, S. J. Moura, J. C. Forman, and H. K. Fathy, "Plug-in hybrid electric vehicle charge pattern optimization for energy cost and battery longevity," *Journal of Power Sources*, vol. 196, no. 1, pp. 541–549, Jan. 2011.

[18] T. Zündorf, "Electric Vehicle Routing with Realistic Recharging Models," Master's Thesis, Department of Informatics, Karlsruhe, Germany, Nov. 2014.