A Fast Electric Vehicle Planner Using Clustering

Jaël Champagne Gareau, Éric Beaudry, and Vladimir Makarenkov

Abstract Over the past few years, several studies have considered the problem of Electric Vehicle Path Planning with intermediate recharge (EVPP-R) that consists of finding the shortest path between two given points by traveling through one or many charging stations, without exceeding the vehicle's range. Unfortunately, the exact solution to this problem has a high computational cost. Therefore, speedup techniques are generally necessary (e.g., contraction hierarchies). In this paper, we propose and evaluate a new fast and intuitive graph clustering technique, which is applied on a real map with charging station data. We show that by grouping nearby stations, we can reduce the number of stations considered by a factor of 13 and increase the speed of computation by a factor of 35, while having a very limited trade-off increase, of less than 1%, on the average journey duration time.

Keywords Electric vehicles · Charging stations · Planning · Clustering · Graphs

1 Introduction

Electric vehicles (EVs) are an attractive alternative to fossil-fuel vehicles to reduce air pollution. However, their limited range and their high-charging time represent a major obstacle to their massive adoption Smart and Schey (2012). Moreover, long journeys require careful planning to determine the charging stations to be used in order to avoid running out of energy. For example, an average EV currently has a range of around 250 km and needs to make many recharging stops on long journeys.

J. Champagne Gareau (✉) · É. Beaudry · V. Makarenkov
Université du Québec á Montréal, P.O. Box 8888, H3C 3P8, QC, Montréal, Canada
e-mail: champagne_gareau.jael@courrier.uqam.ca

É. Beaudry
e-mail: beaudry.eric@uqam.ca

V. Makarenkov
e-mail: makarenkov.vladimir@uqam.ca

T. Chadjipadelis et al. (eds.), *Data Analysis and Rationality in a Complex World*, Studies in Classification, Data Analysis, and Knowledge Organization,
https://doi.org/10.1007/978-3-030-60104-1_3

EV path planning with intermediate recharges (EVPP-R) is a complex problem which cannot be effectively solved by conventional approaches because one needs to consider not only the different variables applicable to conventional vehicles (the wind, the energy needed to fight the air resistance relative to the speed, the traffic, eventual detours, etc.), but also aspects specific to EVs, such as the level of charging stations (which influences the charging speed), the non-linearity of the charging curve of the battery, the topography of the map (EVs can recover some energy when moving downhill) as well as the expected waiting time at the charging station.

In addition to these considerations, a non-negligible characteristic of a good EV planner is its running time. Many well-known techniques are commonly used to accelerate the graph search, including graph contraction hierarchies (Geisberger et al. 2012) and various search heuristics. However, no one has yet tried to decrease the journey computation time by using clustering techniques in order to decrease the number of nodes considered in the graph. *The main contributions of our paper are as follows*:

- a fast and intuitive clustering technique to solve the EVPP-R problem;
- evaluation of the proposed technique on real data to assess its performance.

An EV planning framework that takes into account the fact that the EV recharges itself during a journey (by using regenerative braking) already exists (Sachenbacher 2011). However, it does not take into account the possibility of using charging stations to charge the EV battery midway. The algorithm used in Sachenbacher (2011) is a variant of A* (Hart et al. 1968), called *Energy-A**. This method allows one to find a solution in $\mathcal{O}(n \log n)$, where n is the number of nodes of the graph. Techniques that tackle the problem of midway charging have also been proposed. Some of them consider the problem of a single EV path planning (what we call EVPP-R), while others consider the problem of an EV fleet routing (EVRP). In this article, we focus on the former. The two main approaches which have been used to solve the EVPP-R problem are Dijkstra/A*-based (Baouche et al. 2014, Champagne Gareau 2018), or dynamic programming/MDP-based (Sweda and Klabjan 2012) algorithms. While these techniques try to minimize the sum of the travel time and charging time, some new techniques also consider the expected waiting time (Champagne Gareau 2019). The majority of Dijkstra/A*-based techniques use heuristic graph searches as well as contraction hierarchies (Geisberger et al. 2012) in order to accelerate the computations. In this paper, we show that clustering is another way to speedup the computation (that can be combined with the other speedup techniques).

2 Problem Formulation and Base Planner

We begin by presenting how we model the problem, including the road network representation, the problem definition, and what we consider an optimal solution.

Definition 3.1 A *road network* M is modeled by a tuple $(V, E, \lambda, \sigma, S)$, where (V, E) is a digraph and λ, σ are two labelings of the edges. More specifically:

- V is the set of nodes (latitude, longitude) on the map;
- E is the set of road segments (edges);
- $\lambda : E \to \mathbb{R}^+$ gives the length (in m) at every edge;
- $\sigma : E \to \mathbb{R}^+$ gives the expected speed (in m/s) at every edge;
- S is the set of all charging stations.

Every charging station $s \in S$ is associated to the nearest vertex $v_s \in V$. The σ labeling can be based on empirical data of the mean speed on every edge, or can simply be the maximum allowed speed of each road segment.

Definition 3.2 An *EVPP-R* problem is defined by the tuple $(M, \rho, \alpha, \omega)$, where

- M is the road network;
- $\rho \in \mathbb{R}^+$ is the range of the EV;
- α and ω are the points of departure and arrival.

Remark 3.1 We assume that $\alpha, \omega \in V$. If it is not the case, a KD-Tree can simply be used to find the nearest corresponding nodes in the graph.

Definition 3.3 A *solution* of an EVPP-R problem $(M, \rho, \alpha, \omega)$ is a tuple (P, Q), where

- P is a finite sequence $(P_i)_{i=0}^{k}$ (where $P_i \in V$);
- Q is a subsequence $(P_{i_j})_{j=0}^{b+1}$ of P containing the P_i's where a station is used;
- $P_{i_0} = P_0 = \alpha$ and $P_{i_{b+1}} = P_k = \omega$;
- $\forall j \in \{0, 1, \dots, b\}$, $\mathrm{d}(P_{i_j}, P_{i_{j+1}})$[1] $\leq \rho$.

In other words, P is the sequence of nodes that the EV needs to travel by according to the solution, and Q is a subsequence of P containing the charging stations that need to be used in the journey (as well as α and ω). Our objective is to find a solution that minimizes the total time of the journey, including the travel time and the charging time (we don't consider the waiting time, but the planner can easily be modified to consider it as well (Champagne Gareau 2019; Sweda et al. 2017)). This is formalized in the next definition.

Definition 3.4 An *optimal solution* to an EVPP-R is a solution (P, Q), as stated in Definition 3.3, which minimizes the objective function $\mathrm{Z}(P, Q) = \mathrm{TT}(P) + \mathrm{CT}(Q)$, where TT is the expected travel time and CT is the expected charging time:

$$\mathrm{TT}(P) = \sum_{i=0}^{k-1} \frac{\lambda(P_i, P_{i+1})}{\sigma(P_i, P_{i+1})} \quad \text{and} \quad \mathrm{CT}(Q) = \sum_{i=1}^{b} \mathrm{ST}(Q_i).$$

$\mathrm{ST}(Q_i)$ is the charging time when considering Q_i's state of charge and level.

[1] $\mathrm{d}(A, B)$ is the distance in the graph between A and B.

We now present our baseline planner. First, the distance between each pair of stations is pre-computed and stored in a matrix $D = (D_{ij})$, where $D_{ij} = \mathrm{d}(S_i, S_j)$. We also store the optimal path between each of them. We then build a simplified graph (s-graph) (V', E') containing the nodes associated with the stations and the edges with weights corresponding to the pre-computed distances between every pair of stations. For every request (α, ω, ρ), we temporarily add α and ω to the s-graph and use Dijkstra's algorithm twice (from α, and from ω on the reversed graph) to add edges from α to every station and from each station to ω. The computations for this step can be accelerated by using contraction hierarchies. The new graph obtained is a complete graph, from which we remove every edge whose length is larger than ρ. We then execute the A* algorithm on this new graph (using the great-circle distance as heuristic) from α to ω, which is enough to get a sequence Q, as specified in Definition 3.3. This sequence satisfies the last part of Definition 3.3 insofar as every intermediary node is a charging station. Consequently, there is no need to consider the range of the vehicle at this point. The sequence P can then be found from Q using the previously computed paths.

The base planner has a time complexity of $\mathcal{O}(|V| \log |V| + |E|)$, equivalent to Dijkstra's algorithm's complexity. In a real road network, the maximum number of intersections at a given node is bounded by a small constant (i.e., $|E| \in \mathcal{O}(|V|)$). This implies that we can simplify the time complexity of the algorithm to $\mathcal{O}(|V| \log |V|)$. In the next section, we show how to improve the aforementioned algorithm by clustering the charging stations before creating the s-graph.

3 Clustering

To implement a faster EV planning algorithm, we propose to cluster the stations. We introduce the parameter $d_{\max} \in \mathbb{R}^+$ corresponding to the maximal distance between the center of a cluster and the stations it contains. We then create the clusters as described in Algorithm 1, which has the time complexity of $\mathcal{O}(K(|S|^2 + |V|))$, where K, $|S|$ and $|V|$ are, respectively, the number of clusters, stations and nodes in the graph. $|S|^2$ is currently small compared to $|V|$, but is expected to grow in the future.

Algorithm 1 Fast charging stations clustering

```
1: Find the two nearest charging stations $s_1, s_2 \in S$        ▷ using the pre-computed matrix
2: while $\mathrm{d}(s_1, s_2) \le d_{\max}$ do
3:     Find the midway node $m \in V$ between $s_1$ and $s_2$        ▷ using Dijkstra from $s_1$ to $s_2$
4:     Find $C = \{s \in S | dist(s, m) \le d_{\max}\}$        ▷ using Dijkstra from $m$
5:     $S \leftarrow (S \setminus C) \cup \{m\}$        ▷ $m$ is the representative of the new cluster
6:     Find the two nearest charging stations $s_1, s_2 \in S$
```

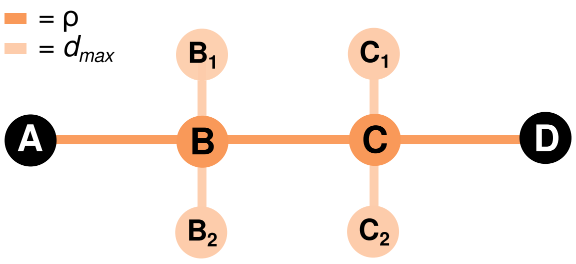

Fig. 1 The distance between clusters B and C is equal to the range ρ. To reach C_1 or C_2 from B_1 or B_2, one needs to travel $\rho + 2d_{\max}$ km. Alas, the planner will consider only the distance between the centers B and C

When we use the base planner described previously along with the clustering, some previously feasible paths may become unfeasible because the new itinerary will always pass through a cluster's center before reaching one of its stations. To solve this problem, the range considered by the algorithm needs to be modified. Figure 1 gives an instance of this problem, and Proposition 3.1 gives the solution.

Proposition 3.1 *Let ρ be the true range of the EV and ρ' be the range considered by the planner. To always have a feasible plan, ρ' needs to be set to $\rho - d_{\max}$ between α and the first traversed cluster, and between the last traversed cluster and ω. Between pairs of clusters, it needs to be set to $\rho - 2d_{\max}$.*

Proof Let $\{c_1, \ldots, c_k\}$ be the traversed clusters in the path from α to ω returned by the planner. The path from α to c_1 is less than or equal to ρ'. Since c_1 is the center of a cluster[2] and the stations have at most a distance of $d_{\max}$ from the center, the range of the EV needs to be $\rho = \rho' + d_{\max}$, so the considered range must be $\rho' = \rho - d_{\max}$. The same argument applies to the segment of the plan between the last cluster used and the arrival point. Similarly, for the path between $s_i \in c_i$ and $s_{i+1} \in c_{i+1}$, we have

$$\mathrm{d}(s_i, s_{i+1}) \le \mathrm{d}(s_i, c_i) + \mathrm{d}(c_i, c_{i+1}) + \mathrm{d}(c_{i+1}, s_{i+1}) \le d_{\max} + \rho' + d_{\max}$$

In some cases (e.g., Fig. 1) we have equality, so the bound is tight. □

Proposition 3.1 presents the theoretical worst case. However, in the implementation, it is possible to consider the *radius* r_i of every cluster c_i (the maximal distance between the center and a station inside it). The range ρ' that needs to be considered between pairs of clusters (c_i, c_j) can then be set to $\rho' = \rho - r_i - r_j \le \rho - 2d_{\max}$.

The new range ρ' can make some previously feasible paths not found by the planner. While at first it may seem to be an unacceptable compromise, a simple solution is to compute the journey using clustering, and if the journey is not found because of the new range, recompute the journey without clustering. In the evaluation of our technique, we call this strategy the *amortized version* of our planner.

Algorithm 2 shows how the different steps fit together in the complete planner. Note that another version of the problem starts with the s-graph given as input, in which case lines 3, 5, and 10 can be omitted. Nevertheless, even when the s-graph

[2]For the sake of simplicity, c_i denotes both the set of stations in the cluster and the center node.

Algorithm 2 Complete planner algorithm

```
1: Compute the matrix $D$ and the optimal path between every pair of stations
2: Compute the clusters using Algorithm 1
3: Construct the s-graph containing the representative of each cluster
4: for all request $(\alpha, \omega, \rho)$ do
5:     Run Dijkstra's algorithm from $\alpha$ (on the original graph) and $\omega$ (on the reversed graph)
6:     Add $\alpha$ and $\omega$ to the s-graph and add edges with length $\leq \rho$
7:     Run the A* algorithm on the s-graph from $\alpha$ to $\omega$ to find the sequence $Q$
8:     if the path is infeasible then                    ▷ When we use the amortized strategy
9:         Run A* on the s-graph (without clustering)
10:    Find the sequence $P$ from $Q$ using the already computed paths
```

needs to be constructed, line 5 is done in the original graph, so the time complexity of this step is not affected by the clustering (but can be accelerated using contraction hierarchies). Furthermore, lines 6 and 10 have a negligible time compared to the others. Therefore, we focus on the time complexity of lines 7–9 in the evaluation.

When we use clustering ($d_{\max} > 0$), we can compute P (Line 10) as in the previous section. Note, however, that Q now contains the clusters that the EV will pass through on its journey (instead of containing raw stations), but gives no information about which station to select inside the cluster. This choice is made at runtime (i.e., a short time before the vehicle arrives at a decision point before the cluster) rather than during the initial planning. Therefore, clustering gives the opportunity of considering real-time data (e.g., road conditions, charging stations occupancy state, etc.) to select the most suitable station inside the next cluster on the path.

4 Empirical Evaluation

Our algorithm was evaluated using real data from the Province of Québec, Canada. The map data (i.e., the nodes and the road segments) were extracted from the Open-StreetMap project. We chose this map in our tests because the territory is vast, the journeys between pairs of cities can be very long and the network of stations is relatively well developed. The graph we generated from these data had 2 923 013 vertices and 5 907 672 edges. The stations considered in our tests were real stations from this territory. Our dataset included 1162 charging stations (Level 2 and 3). We considered all stations in the data as if they were Level 3 because EV planners are primarily used for a long itinerary where fast charging is a must and there were not enough Level 3 stations in the dataset to test our algorithm adequately (only 122).

To evaluate our method, we generated 1000 requests consisting of the departure α, arrival ω (both chosen at random in the graph) and EV range ρ (generated uniformly between 90 and 550 km, based on the most common EV ranges available on the market). The optimal solution length ranged from 150 to 1000 km. Every generated request required at least one stop at a station for the EV to be able to reach the destination. Our tests consisted of running these 1000 requests by our planner on

Table 1 Empirical results obtained for 1162 real stations in Québec (Canada)

Problem Param.	Clust. Param.		Base version		Amortized version	
$d_{\max}$ km	C #	JDIR %	FR %	CPU ms	FR %	CPU ms
0.0	1162	0.0	0.0	26.56	0.0	26.56
2.5	487	0.0	0.0	3.385	0.0	3.385
5.0	342	0.2	0.4	1.541	0.0	1.647
10.0	236	0.2	0.8	0.588	0.0	0.801
15.0	188	0.6	0.9	0.523	0.0	0.762
20.0	150	1.0	1.4	0.382	0.0	0.754
30.0	111	2.3	2.0	0.265	0.0	0.796
40.0	87	2.8	8.2	0.226	0.0	2.404

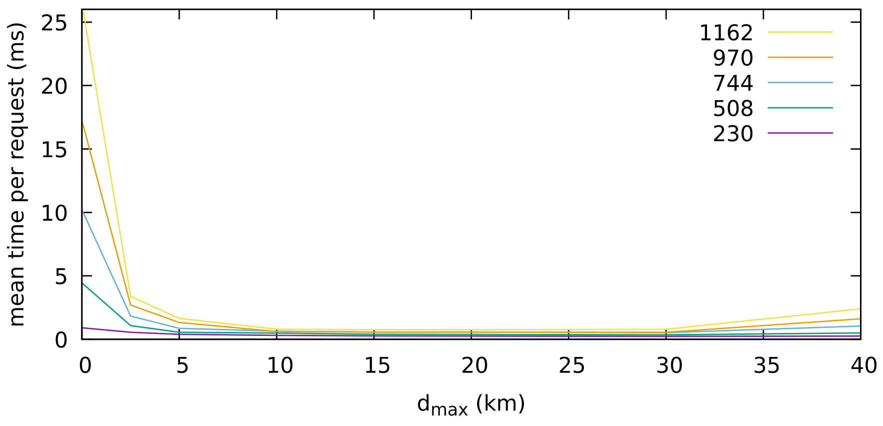

Fig. 2 Mean CPU time for different density of stations using amortized strategy

an Intel Core i5 7600k processor. We compared the results obtained with different combinations of parameters: different subsets of stations (20, 40, 60, 80, or 100% of the total) to measure the effect of stations density on clustering, and different values of the parameter $d_{\max}$ (0, 2.5, 5, 10, 15, 20, 30, and 40 km). For the sake of simplicity, it was assumed in the tests that $\sigma(e) = 90$km/h for all $e \in E$ and that $ST(s) = 30$ min for all $s \in S$ since it doesn't influence the clustering efficiency.

Table 1 presents the results obtained by running the tests described previously. The columns denote, respectively, the parameter $d_{\max}$, the number of clusters (**C**), the Journey Duration Increase Rate (**JDIR**), the journey Failure Rate (**FR**), and the average computation time (**CPU**) (with and without amortized strategy). When considering the columns $d_{\max}$ and C, we observe that even a small value of $d_{\max}$ reduces drastically the number of clusters (since they are merged). The percentage of remaining stations decreased from 58.1% (with the smallest $d_{\max}$) to 92.5%.

When using the amortized strategy, the majority of the performance improvement is preserved, while eliminating the infeasible paths (as can be seen by comparing the two pairs of columns (FR, CPU) and by looking at Fig. 2). Column JDIR shows that the use of our clustering technique can slightly increase the duration of the journeys returned by the planner. The running time increase is proportional to $d_{\max}$.

Finally, by comparing the different curves in Fig. 2, we can conclude that when the number of stations is larger (i.e., the density of stations on the map is higher), the decrease in the running time due to clustering is also larger. Thus the usefulness of our clustering technique will increase while more stations are getting installed. Our results suggest (when considering the columns JDIR and CPU (Amortized)) that the optimal $d_{\max}$ value for our dataset is between 15 and 20 km. A larger value of $d_{\max}$ results in an increase of more than 1% of the average journey duration and an increase in the failure rate which is large enough to mitigate the time saving (i.e., the CPU time starts to increase when $d_{\max}$ becomes larger than 20).

5 Conclusion

In this paper, we proposed to use a fast graph clustering technique to reduce the running time of a planner solving the EVPP-R problem. Our results show that clustering of charging stations allows for a factor 13 decrease in the number of stations to be considered, and a factor 35 decrease in the running time in the simplified graph, while having no decrease in the number of feasible journeys (with the amortized strategy). The main advantages of our technique are its simplicity, intuitiveness and computational efficiency, whereas its main disadvantage is a limited trade-off of 1% increase to the average journey duration time. As future work, we plan to investigate if the use of traditional graph clustering techniques such as MCL and k-medoids on graph spectral embeddings, which are slower than the proposed algorithm, could improve some of our results.

Acknowledgements This work was supported by the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and by the *Fonds de recherche du Québec—Nature et technologies* (FRQNT).

References

Baouche, F., Billot, R., Trigui, R., El Faouzi, N.E.: Electric vehicle green routing with possible en-route recharging. In: International Conference on Intelligent Transportation Systems (ITSC), pp. 2787–2792 (2014)

Champagne Gareau, J., Beaudry, É., Makarenkov, V.: Planification d'itinéraires quasi-optimaux pour un véhicule électrique en considérant le regroupement de bornes de recharge et leur probabilité d'occupation. In: XXV-émes Rencontres de la Société Francophone de Classification (SFC2018), pp. 5–8. Paris, France (2018)

Champagne Gareau, J., Beaudry, É., Makarenkov, V.: An efficient electric vehicle path-planner that considers the waiting time. In: Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. ACM, Chicago, United States (2019)

Geisberger, R., Sanders, P., Schultes, D., Vetter, C.: Exact routing in large road networks using contraction hierarchies. Transport. Sci. **46**(3), 388–404 (2012)

Hart, P., Nilsson, N., Bertram, R.: A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Man Cybern. A. Syst. Humans - TSMCA **4**(2), 100–107 (1968)

Sachenbacher, M., Leucker, M., Artmeier, A., Haselmayr, J.: Efficient energy-optimal routing for electric vehicles. In: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI), pp. 1402–1407 (2011)

Smart, J., Schey, S.: Battery electric vehicle driving and charging behavior observed early in the EV project. In: SAE Technical Papers, pp. 27–33 (2012)

Sweda, T.M., Dolinskaya, I.S., Klabjan, D.: Adaptive routing and recharging policies for electric vehicles. Transport. Sci. **51**(4), 1326–1348 (2017)

Sweda, T.M., Klabjan, D.: Finding minimum-cost paths for electric vehicles. In: IEEE International Electric Vehicle Conference (IEVC), pp. 1–4 (2012)