Cooperative Electric Vehicles Planning

Jaël Champagne Gareau
Université du Québec à Montréal
Montréal, Québec, Canada
champagne_gareau.jael@uqam.ca

Marc-André Lavoie
Cégep du Vieux-Montréal
Montréal, Québec, Canada
malavoie@cvm.qc.ca

Guillaume Gosset
Université du Québec à Montréal
Montréal, Québec, Canada
gosset.guillaume@courrier.uqam.ca

Éric Beaudry
Université du Québec à Montréal
Montréal, Québec, Canada
beaudry.eric@uqam.ca

ABSTRACT

This paper introduces the Cooperative Electric Vehicles Planning Problem (CEVPP), which consists in finding a path for each vehicle of a fleet of electric vehicles, such that the global plan execution time (including travel time, charging time and waiting time) is minimal (e.g., by limiting the number of vehicles who need to charge simultaneously at the same charging station, which leads to waiting time). We show that the strategy which consists in planning each possible permutation of EVs and keeping the one providing the best solution is not only time intractable, but also not optimal. We propose different centralized planning algorithms to solve CEVPP instances: (1) a baseline non-cooperative CEVPP planner, (2) an optimal cooperative planner that finds a solution inside a carefully designed state space, and (3) multiple variants of an approximate cooperative planner based on the Cooperative-A* algorithm. We compare the solutions' quality and computation times obtained by these CEVPP planners. Our empirical results show that our best approximate cooperative EV planner found solutions with a reasonably small computational overhead compared to the baseline algorithm. The solutions found by our cooperative planners had significantly lower plan execution time globally, including travel time, waiting time and charging time, than the solution found by our baseline non-cooperative planner. On average, our empirical results show that our cooperative algorithms decreased the global (including each EVs) waiting time by more than 90%, while having a negligible impact on the charging and driving time.

CCS CONCEPTS

• **Mathematics of computing** → **Graph algorithms**; Combinatorial algorithms; • **Theory of computation** → **Shortest paths**; • **Computing methodologies** → **Cooperation and coordination**; **Discrete space search**; **Heuristic function construction**; *Multi-agent planning*; *Planning for deterministic actions.*

KEYWORDS

Electric Vehicles; Planning; Waiting Time; Global Optimization; Cooperation; Occupation; Charging Stations

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

ACM Reference Format:
Jaël Champagne Gareau, Marc-André Lavoie, Guillaume Gosset, and Éric Beaudry. 2024. Cooperative Electric Vehicles Planning. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

To fight climate change, it is important to gradually shift away from fossil fuels towards greener energy sources, particularly in transportation. Electric Vehicles (EVs) offer a sustainable alternative to fossil-fuel vehicles and have become increasingly widespread in many countries due to improvements in their range and the availability of more charging stations. However, as the battery capacity of EVs has increased, so has the time it takes to charge them. This leads to longer waiting times when charging stations are occupied, which is a barrier to the widespread adoption of EVs. The increasing number of EVs on the road and the longer charging times are both factors that contribute to the longer waiting times.

In the last decade, many researchers have examined the planning problems associated with electric vehicles. These problems can be divided into two main categories. The first category is path planning for a single EV, often referred to as EVPP (Electric Vehicles Path-Planning) [2, 8]. This problem focuses on EV-specific factors, such as regenerative braking and the location of charging stations on the road map. The second category is routing for a fleet of EVs, known as EVRP (Electric Vehicles Routing Problem) [5, 13]. In this case, the goal is to find the most efficient way to dispatch a group of EVs, such as those used by a delivery company.

In recent years, some electric vehicles planning algorithms have begun to consider waiting time as an additional factor to optimize, in addition to the more commonly considered travel time and charging time [4, 10, 20]. These algorithms aim to minimize the journey for one specific EV at a time. However, in real-world situations where multiple EVs have predetermined destinations, it is desirable to plan their routes collectively so that we can prevent multiple EVs from needing to charge at the same station at the same time. For instance, some EVs could be directed to less congested charging stations, even if it necessitates a marginally longer journey. Finding global plans can significantly reduce overall waiting time and improve the experience for EV users. This problem has recently garnered attention in the research community, being referred to as an important issue:

> "An open challenge is to devise algorithms for socially optimal real-time routing with a reasonable response time for a large number of vehicles." [1]

In this paper, we introduce the Cooperative Electric Vehicles Planning Problem (CEVPP), which aims to find a plan that minimizes the overall journey time of a fleet of electric vehicles, rather than planning for each vehicle independently. The aforementionned (multi-agent) EVRP has some important differences compared to the proposed CEVPP. The former focuses on a fleet of (e.g., delivery) EVs controlled by the same entity. The vehicles start and end at the same depot. The objective is to find a minimum set of EVs able to complete all tasks with minimal cost (travel time + energy). In the latter, each EV is owned by a different end-user and their respective journey can start and end at different positions. The problem is dynamic with new EV requests continually arriving. The objective is to minimize the total (traveling, charging and waiting) time.

Our main contributions in this paper are as follows:

- we introduce a new EV planning problem: *CEVPP*;
- we propose an optimal and an approximate algorithm to solve CEVPP instances;
- we compare the proposed algorithms on real-world networks, considering the computation time and solution quality, using a non-cooperative planner as a baseline.

The rest of the paper is organized as follows: In Section 2, we discuss relevant cooperative algorithms. Section 3 provides a mathematical formulation of CEVPP. In Section 4, we present each of the proposed CEVPP planners. Section 5 presents the evaluation of our methods, including the testing methodology, results, and analysis. Finally, in Section 6, we provide our conclusion and present some ideas of future works.

2 RELATED WORKS

To the best of our knowledge, there is no work about cooperative planning in the context of electric vehicles. However, many works in the field of automated planning and computer games have proposed cooperative multi-agent algorithms used in other contexts [12, 18, 19]. We present some of them in this section, from which some of our proposed algorithms are based on.

One simple cooperative planning technique is *Local-Repair A** (LRA*) [23], which uses the A* algorithm [7] to find the optimal path of each agent (non-cooperatively). During execution of the plans, agents can detect imminent collision in the state space and recompute (i.e., repair) local parts of its plan. This method is simple, but requires frequent recomputations. Furthermore, it does not find globally optimal solutions.

*Cooperative-A** is another cooperative planning algorithm [17]. Similar to LRA*, it uses A* to compute the path of each agent one-by-one. However, after finding the path of an agent, its position at relevant times is recorded into a reservation table. Subsequent paths found are computed while taking into account the reservation table, which allows the agents to get around the positions occupied by already computed agents at each time step. This leads to an implicit priority of agents whose paths are computed first over those whose paths are computed after. Since the order for which the agents' path are computed directly impact the solution obtained, different orderings have been proposed [11]. Assuming there are k agents, some possible orderings are: (1) first arrived, first computed (1 order); (2) first departure time, first computed (1 order) (3) every permutation ($k!$ orders); (4) cascading insertion order ($\frac{k(k-1)}{2}$ orders); (5) random order (H orders, where H can be chosen to control a tradeoff between computation time and solution quality). One problem of Cooperative-A* is that many orders might be equivalent, and therefore we might waste a lot of time and resources during computation. The M* algorithm [21] partitions the state space into independent subspaces such that agents in one subspace have a minimal number of collisions with agents in other subspaces. This allows planning inside each subspace independently, and therefore the planning time is greatly decreased. However, when we can expect many collisions, the running time quickly increases and the algorithm becomes intractable.

The previously mentioned multi-agent cooperative planning algorithms are either suboptimal, or optimal but intractable in practice. They are also difficult to adapt to the context of electric vehicles, mostly for two reasons. First, the mentioned algorithms each consider *hard collisions* (i.e., two agents can't share the same position at the same time), whereas EV collisions are *soft collisions* (an EV can pass by a station without needing to stop, and multiple EVs can be at the same station at the same time, incurring a waiting time penalty, but no failure). Second, the mentioned algorithms are designed to consider the position of agents in the state space at discrete time steps, which makes sense in a grid, but not on a road network where the EVs all have different speeds and continuous travel, waiting and charging times.

For EVs to plan their respective path accurately, it is important to predict the waiting times at the different possible charging stations. Recently, some studies have tackled this problem. Some of them propose a supervised machine learning model to estimate charging stations waiting time, including idle time, i.e., the time an EV remains parked after charging is complete [3, 14]. Others have proposed using historical occupancy data to estimate station availability and waiting time depending on the time and day of arrival [4, 15].

3 COOPERATIVE ELECTRIC VEHICLES PLANNING PROBLEM

This section formally defines the concepts related to CEVPP. The notions of road networks, EV requests, CEVPP, and solutions are respectively defined next.

Definition 1. A *road network* M is modeled by a tuple (V, E, λ, μ, S), where (V, E) is a directed graph and λ, μ 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);
- $\lambda\colon E \to \mathbb{R}^+$ gives the length (in m) of every edge;
- $\mu\colon E \to \mathbb{R}^+$ gives the expected speed (in m/s) at every edge;
- $S \subseteq V$ is the set of all charging stations.

Remark 1. The labelings λ and μ defined in Definition 1 can be used to create a labeling that specifies the expected time needed to traverse an edge, using edges that represent time rather than distance in the graph. Both formulations are interchangeable. The μ labeling can be derived from empirical data on the average speed

of vehicles on each edge or can be set to the maximum allowable speed of each road segment.

Remark 2. The model currently considers homogeneous charging stations, each of them having a single charging port, but it can easily be extended to consider heterogeneous stations (e.g., different number of charging ports, different charging rate, etc.). In the case of charging rate, no algorithmic change would be necessary in the proposed algorithms (only the formula to compute charging time would change).

Definition 2. An *EV request* is a tuple $(\alpha, \omega, \rho, \tau)$ where:

- $\alpha \in V$ is the departure node;
- $\omega \in V$ is the destination node;
- $\rho \in \mathbb{R}^+$ is the range (battery autonomy in m) of the vehicle;
- $\tau \in \mathbb{R}^+$ is the departure time.

Remark 3. In addition to the range, the previous definition can also be extended to include other characteristics of EVs, such as travel and charging speed. These additionnal variables can be included in the proposed planning algorithms with only trivial modifications.

Definition 3. The *Cooperative Electric Vehicles Path-Planning Problem* (CEVPP) is defined by the tuple (M, R) where:

- M is the road network;
- $R = \langle(\alpha_1, \omega_1, \tau_1, \rho_1), \ldots, (\alpha_k, \omega_k, \tau_k, \rho_k)\rangle$ is a list of EV requests in an arbitrary order.

Definition 4. A *solution* to a CEVPP instance (M, R) is a global plan, consisting of a set of itineraries (one for each request $r \in R$). More precisely, a solution is denoted

$$\begin{aligned}\pi &= \langle \pi_1, \pi_2, \ldots, \pi_k \rangle \\ &= \langle \langle v_{11}, v_{12}, \ldots, v_{1n_1} \rangle, \langle v_{21}, v_{22}, \ldots, v_{2n_2} \rangle, \ldots, \langle v_{k1}, v_{k2}, \ldots, v_{kn_k} \rangle \rangle,\end{aligned}$$

where for all $i \in \{1, 2, \ldots, k\}$, $v_{i1} = \alpha_i$ and $v_{in_i} = \omega_i$. The set of all solutions π is denoted Π.

Many objective functions can be chosen in multi-agent planning. For example, we can try to minimize the longest itinerary (makespan), or the sum of the time of completion of all itineraries (flowtime). However, none of these is ideal, since that may force an end-user of the system to do a large detour of, e.g., 2h, if it helps many other EVs save 5 minutes. Since in the context of CEVPP, the cooperation among EVs should not be made disproportionately at the expanse of one of them, we instead propose to use quadratic deviations. Definitions 5 and 6 present respectively how we measure the cost of a plan and what we will consider an optimal solution.

Definition 5. We define the *cost* of a solution $\pi = \langle \pi_1, \pi_2, \ldots, \pi_k \rangle$ to be $C(\pi) = \sum_{i=1}^{k} C(\pi_i)$, where $C(\pi_i)$ is the sum of traveling, charging and waiting time that the i^{th} EV incurs when following his plan π_i.

Definition 6. An *optimal solution* $\pi^\star$ is a solution in Π that minimizes the sum of the quadratic deviations between the cost of an EV plan $C(\pi_i)$ and its theoretical optimal cost $C^\star(\pi_i)$ (where $C^\star(\pi_i)$ is the cost of the local plan when assuming the EV is alone on the road network, i.e., the waiting time is zero at every station). More precisely,

$$\begin{aligned}\pi^\star &= \arg\min_{\pi \in \Pi} P(\pi) \\ &:= \arg\min_{\pi \in \Pi} \left[\frac{1}{k} \sum_{i=1}^{k} \left(C(\pi_i) - C^\star(\pi_i) \right)^2 \right].\end{aligned}$$

we call $P(\pi)$ the (global) penalty incurred by a plan π.

In practice, a planner in the context of CEVPP would be used in the following way: A central authority would collect and group incoming EV requests until some time limit T or until some predetermined number of requests is reached, and then find a solution to the CEVPP instance corresponding to the batch of EV requests. When a new batch of EV requests is ready to compute, the EVs from the previous batch that have not yet reached their destination can be included, so that the new and the old EVs in the system can both consider each other from now on. This also improves robustness of the plans by addressing uncertainties indirectly: if a driver takes a break or other unexpected changes happens, the computation of the next batch will account for this.

4 PROPOSED PLANNERS

In this section, we present a simple non-cooperative planner, which we use as a baseline to measure the global plan quality improvement that cooperative planning allows, and two cooperative EV planners, the former being optimal (Exhaustive-Search planner) and the latter being approximate (a permutations-based planner).

To simplify the presentation of the following algorithms, we assume that we have first constructed a graph containing the charging stations (which we will call the stations' graph, or *s-graph*), and where an arc (s_i, s_j) is present in the graph if there exists a path on the road network going from station s_i to station s_j. Such a graph can be constructed easily by e.g., running Dijkstra's algorithm from each charging stations. Other techniques to create this graph faster can also be used, e.g., graph contraction hierarchies [6], but the time it takes to build this graph is negligible compared to the time it takes to find a cooperative solution, and hence it is not an issue in this research.

We also assume for simplicity in the sections below that the departure and destination of each EVs is a charging station (i.e., $(\alpha, \omega) \in S^2$). Again, this is not an issue in practice because algorithms and graphs can easily be adapted (e.g., nodes representing the departure and arrival of an EV can be added to the s-graph).

We assume below that the A* algorithm is modified such that an arc is only considered if its length is smaller than the range of the EV currently considered. Note that for each mentioned planner, when the A* algorithm is used to find the plan of a specific EV request, it is possible to use instead a more specialized EV path-planning algorithm that considers, e.g., regenerative-braking, such as the Energy-A* algorithm [16].

4.1 Baseline: Non-Cooperative Planner

Our baseline non-cooperative planner is based on the A* algorithm and is presented in Algorithm 1. Basically, each EV request is planned independently (i.e., without considering the other EVs) using the A* algorithm. After each path is planned, we compute the global penalty incurred by the plan found. In the case of the baseline

planner, since every EV takes the geographically shortest path, the penalty is entirely due to waiting times of EVs trying to charge at the same stations. The time complexity of this algorithm is given by $\Theta(k \cdot |S|^2)$, since we run the A* algorithm k times, and this algorithm has a worst time complexity of $\Theta(|S| \log |S| + |S|^2) = \Theta(|S|^2)$ (the number of edges is $\Theta(|S|^2)$ since the s-graph is a complete graph).

Algorithm 1 Baseline Non-Cooperative EV Planner

```
1: procedure ncEVP((M, R = ⟨r_1, ..., r_k⟩): CEVPP)
2:     for all r_i ∈ R do
3:         ▷ Considers travel and charging, but not waiting time
4:         π_i ← A*(M, r_i)
5:         π ← π ∪ {π_i}
6:     Compute the global penalty P(π)
```

4.2 Exhaustive-Search Cooperative Planner

One way to find an optimal solution is to use a graph planning algorithm on a graph that represents the entire state space of a CEVPP instance. The difficulty is in defining the state space in such a way that we have all the information needed in the state variables, but nothing superfluous. Definition 7 presents what we will consider a state. Basically, a state contains for each EV its position (i.e., the id of the current charging station) at a specific time t, and its planned time of departure from that station.

Definition 7. We define a *state* to be an array

$$\sigma = [(\sigma_1^s, \sigma_1^t), (\sigma_2^s, \sigma_2^t), \ldots, (\sigma_k^s, \sigma_k^t)],$$

where:

- σ_i^s is the charging station currently used by the EV i;
- σ_i^t is the planned departure time of EV i from station σ_i^s.

Algorithm 2 presents the main steps of the Exhaustive-Search Cooperative EV Planner (escEVP). The planner is based on the A* algorithm and uses the state space defined above. The initial state is simply built by copying the starting position and departure time from the set of EV requests R. Then, at each considered state, we check if the state is terminal (i.e., each EV i is at its destination node ω_i). If not, we add to the priority queue *open* the states reachable from the current state σ. Since at each state transition, we only make one EV move, we create a new state for each station that each EV i can reach from its current station σ_i^s considering its range ρ_i. The waiting time and charging time are considered in the ComputeTimeDeparture() function. This function checks at current state σ if there is already an EV j charging at station s (i.e., $s = \sigma_j^s$). If so, it considers the time at which j will leave the station (given by σ_j^t) and uses that to compute its own departure time $\sigma_i'^t = \sigma_j^t + \text{ChargingTime}(i, s, \sigma)$, where the charging time is simply computed by considering the range of the EV and the distance it traveled since its last charge. When a new state is constructed, we compute its heuristic cost (used as priority in the queue *open*), which is the minimum for each EV of $f_i = g_i + h_i$, where $g_i = \text{Cost}(i, \sigma')$ and $h_i = \text{Heuristic}(i, \sigma', r_i)$. The cost g_i is simply given by the difference between the initial departure time τ_i of EV i and the departure time σ_i^t of the current station σ_i^s. Finally, h_i is computed with the estimated time to reach the destination node ω_i (using the graph distance from σ_i^s to ω_i). The heuristic h considers both the travel time and the charging time of the vehicle. Since in practice, there might be a detour or some waiting time, the heuristic is a lower bound on the actual time of arrival, and is therefore admissible.

Algorithm 2 Exhaustive-Search Cooperative EV Planner

```
1: procedure escEVP((M, R = ⟨r_1, ..., r_k⟩): CEVPP)
2:     open ← Empty Priority Queue of (state, cost f = g + h)
3:     open.push(InitialState(M, R), 0)
4:     while not open.empty() do
5:         σ ← open.pop()
6:         if IsGoalState(σ) then σ* ← σ; break
7:         for all vehicle i ∈ {1, ..., k} do      ▷ any EV can move
8:             for all s ∈ ReachableStations(σ_i^s, ρ_i) do
9:                 if EV i already visited s then continue
10:                σ' ← σ   ▷ state σ' is same as σ except for EV i
11:                σ'[i] ← (s, ComputeTimeDeparture(i, s, σ))
12:                f ← min_{i∈{1,...,k}} (Cost(i, σ') + Heuristic(i, σ', r_i))
13:                open.push(σ', f)
14:    Extract global plan π from σ*
15:    Compute the global penalty P(π)
```

Assuming the state space Σ only contains the current position $\sigma_i^s \in S$ of each EV i, then the number of possible states is $|\Sigma| = |S|^k$. Since in practice, the state space also contains the planned departure time of each EVs, the true size of the state space is greater. Therefore, the worst-case time complexity of Algorithm 2 is $\Omega(|S|^k)$ and the problem is thus in EXPTIME.

4.3 Permutations Cooperative Planner

Our next cooperative EV planner is inspired by the Cooperative-A* algorithm mentioned before [17]. As for Cooperative-A*, our proposed planner, named Permutations Cooperative EV Planner (pcEVP), computes a plan for each EV one-by-one using the A* algorithm. We use a modified version of A* that records into a reservation table the charging stations and the time of departure from each of them. For the plans of the other EVs computed subsequently, the planner considers the reservation table and therefore considers the waiting time due to stations occupied by already computed plans of prior EVs. The modified A* will in that case either take a detour to avoid the station that would be occupied at the same time, or will conclude that the waiting time is smaller than the detour time and still go to the occupied station.

One important thing to note is that the order in which the EVs are planned will impact the solution's quality. For example, consider the problem in Figure 1. If each EV was alone on the road network, the shortest path of EV α would be $\langle A, B, E, F\rangle$ (total time: 12160 s), and the shortest path of EV β would be $\langle C, B, E, D\rangle$ (total time: 12160 s). Now, let's assume we consider the waiting time. The shortest paths above would lead to two collisions (at stations B and E) and thus lead to some waiting time. If we plan EV β, then EV α, then β will take the shortest path already mentioned. In this case, α will still also take its shortest path. This leads to a waiting time at B (but not at E,

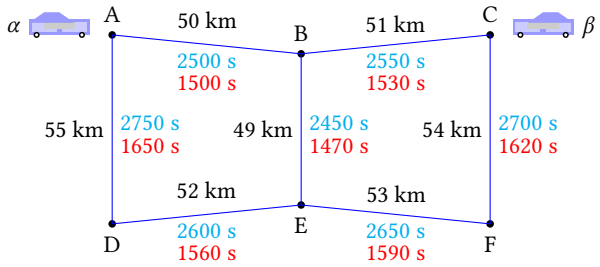

Figure 1: Example of CEVPP where the EV α must move from A to F and the EV β must move from C to D. Each node represent a charging station. We assume the battery of each EV is 12 kWh and the consumption is 200 Wh/km (i.e., their range is 60km). We also assume their speed is constant at 72 km/h and the charging stations have 24 kW power. The numbers in blue represent the travel time, and the numbers in red represent the corresponding charging time.

since α will complete its charging before β arrives). The total time of this global plan is in this case $\max(12160, 12160+1450) = 13610s$, where 1450 is the waiting time of β at B. If instead we plan for EV α first, it will take its shortest path, while β will instead use the path $\langle C, F, E, D\rangle$ to avoid waiting time at station B. There will be waiting time at station E, and the cost of this global plan will be $\max(12160, 12720 + 950) = 13670s$.

We might think that by testing every possible EVs ordering and keeping the smallest cost plan obtained, we will obtain an optimal solution. Alas, it is not the case. As a counterexample, we can see in the above example that none of the two global plans found are optimal. Indeed, the plan

$$\pi = (\pi_\alpha, \pi_\beta) = (\langle A, B, C, F\rangle, \langle C, F, E, D\rangle)$$

does not require any waiting at a charging station and has a cost of $\max(12400, 12720) = 12720$, which is better than the two plans found above by planning EVs one after the other. However, as we'll see in our evaluation, testing multiple orderings still provide in practice a close-to-optimal solution in the vast majority of cases.

If there are k EVs in the CEVPP instance, there are in theory $k!$ possible permutations. It is both impractical (intractable) and unnecessary (many orders are redundant) to find and compare each of the $k!$ possible global plans. Therefore, we will consider in our evaluation many subsets of permutations.

The pseudocode of our approximate cooperative planner, called Permutations Cooperative EV Planner (pcEVP) is shown in Algorithm 3. The algorithm is generic and is the same whatever subset of permutations is returned by GetConsideredPermutations(R). ModifiedA* is simply an A* algorithm that considers (using the reservation table) the waiting time at charging stations were EVs are already committed. The general time complexity of pcEVP is $\Theta(|\mathcal{P}| \cdot |S|^2)$.

One interesting special case of pcEVP is when we only consider the order given implicitly by R (i.e., no permutations). This is not equivalent to the baseline planner (Algorithm 1), because even with a single tested order, pcEVP is making a reservation of the charging stations on a *first planned, first committed* way and therefore considers the waiting time of already committed EVs, whereas the baseline never considers waiting time during planning.

Algorithm 3 Permutations Cooperative EV Planner

```
1:  procedure pcEVP((M, R = ⟨r_1, ..., r_k⟩): CEVPP)
2:      P ← GetConsideredPermutations(R)
3:      C_best ← ∞
4:      for all φ ∈ P do
5:          π ← ∅
6:          R ← Empty Reservation Table
7:          for all r_i ∈ φ do                          ▷ In given order
8:              π_i = ModifiedA*(M, r_i, R)
9:              UpdateReservationTable(R, π)
10:             π ← π ∪ {π_i}
11:         if C(π) < C_best then
12:             π_best ← π
13:             C_best ← C(π)
14:     Compute the global penalty P(π_best)
```

5 EVALUATION

5.1 Test Environment

In this section, we conduct an empirical evaluation to compare the running time and solution quality of the proposed algorithms: (1) a baseline non-cooperative CEVPP planner (Algorithm 1, *ncEVP*), (2) an optimal cooperative planner (Algorithm 2, *escEVP*), and (3) multiple variants of our approximate permutations-based cooperative planner (Algorithm 3, *pcEVP*). The variants we consider are the *ordered by departure* (1 permutation), the *random log(k!)* ($\log(k!) \in \Theta(k \log k)$ permutations sampled randomly among the $k!$ possible permutations) and the *cascade* (k^2 permutations, created by swapping the order of the EVs two-by-two) variants.

The competing algorithms were implemented in C++ and compiled using the GNU g++ compiler (version 12.2). All our experiments were performed on a computer equipped with a 4.2 GHz Intel Core i5-7600K CPU and 32 GB of RAM.

The road network data (i.e., the nodes and the road segments) were taken from the OpenStreetMap project [22]. The territory of the Québec province and the Maritime provinces (Canada) were chosen to carry out our tests. They are vast, the journeys between certain pairs of cities can be very long (thus requiring a large number of recharge) and the network of charging stations is relatively well developed. The former graph generated from these data has

4 416 080 vertices and 8 797 051 edges, while the latter one has 2 105 607 vertices and 4 200 189 edges.

The stations considered in the tests come from a public network of EV charging stations, called the *Electric Circuit* [9]. We use three different sets of stations in our evaluation: one for the Maritimes road network (50 stations), and two for the Québec road network, with respectively 347 and 1816 stations. To simplify, we assume in our implementation that each station has an identical power and that the EVs are charged linearly, recovering a range of 9 km/min.

To assess the performance of the proposed algorithms, we generated random sets of EV requests, ranging from 2 to 1024 EVs per set. For each set, we randomly sampled two stations at least 100 km apart. We then sampled from a 50 km cluster around these stations the departure α and destination ω nodes, to simulate multiple EVs journeying along similar paths. The range of each vehicle was sampled uniformly between 100 and 550 km. Finally, the departure time of each vehicle was sampled uniformly between 0 and 120 min.

5.2 Results

Table 1 reports the detailed results provided by the compared planning algorithms on the three tested road networks. The first two columns present the characteristics of the test (the name of the road network, and the number of simultaneous EVs). The remaining columns account respectively for the penalty $P(\pi)$ (column P) of the solution found and the running time of each compared algorithm (column RT). Each test (i.e., each row in the table) consists in running 50 CEVPP request instance. We report in the table the average over these 50 results. We set a timeout value of 15 minutes. We represent by the symbol '-' the fact that a solver did not manage to solve the 50 CEVPP requests inside a 15-minute period. We also present the results in Figure 2, where the 6 subfigures represent the two evaluated metrics (penalty of the solution and running time) over the three tested road networks.

Note that we do not include the results for the optimal Exhaustive-Search cooperative planner presented in Section 4.2, because that algorithm has an exponential time complexity and did not manage to solve requests within the 15 minutes allotted period. Moreover, since the number of states that need to be considered increase exponentially as the number of EVs increases, the memory consumption is also an issue. For example, tests with 6 EVs on the Maritimes_{50} road network led to a memory consumption that exceeds the hard limit of 24 GB we had set. That being said, we believe this optimal planner to be an important first step in understanding and designing CEVPP solving methods in the future. Based on this algorithm, we plan as future work in designing other optimal CEVPP planners able to prune large parts of the state spaces that would hopefully lead to an optimal planner useful in practice.

When looking at the results, a first surprising insight is that the simplest and fastest cooperative planner, the permutations cooperative planner variant that considers only one permutation (the permutation where we plan vehicles ordered by departure time), managed to find the best solutions (i.e., solutions with the lowest penalty value) in almost all test instances where the number of EVs is at least 32. This is because when there are lots of EVs, there can be a bottleneck at a specific EV charging station. By planning the EVs ordered by departure time, the i^{th} EV knows the stations already reserved and therefore can avoid the bottleneck. It allows to avoid situations where an EV that would have arrived first at a charging station only gets the chance to make a reservation after someone else who needs it less (e.g., who would have arrived 20 minutes after) already reserved it. The random $\log(k!)$ permutations variant has a high likelihood of not considering an order that allows the planner to avoid entirely these bottlenecks (e.g., more than 3 EVs at the same station), whereas the cascade variant will almost never find the optimal order in this case. When the number of EVs is less than 32, those kinds of bottlenecks are significantly less likely, and therefore the advantage of the first variant disappears.

We can summarize the preceding observations by saying that the first variant (ordered by departure time) is best when there are many EVs who tries to charge at the same station (i.e., one large bottleneck), which happens most often when the number of EVs is large compared to the number of stations in the road network, whereas the other two variants are best when there are many small bottlenecks (e.g., many charging stations where 2-3 EVs try to charge at the same time).

Between the *cascade* and the *random* $\log(k!)$ variants, there is no clear winner when considering the solution quality obtained on all three tested road networks. However, since the *random* $\log(k!)$ variant has a $\Theta(k \log k \cdot |S|^2)$ time complexity, whereas the *cascade* variant has a $\Theta(k^2 \cdot |S|^2)$ time complexity, it managed to solve more difficult instances of the problem. Therefore, the former seems more promising. As for the running time of the baseline non-cooperative planner and the *Ordered by departure* variant of our cooperative permutations planner, we see in the results that they only differ by a constant, which is expected, since the only overhead is the cost reordering the EV requests and of managing the reservation table.

In Table 2, we showcase the average percentage reduction in penalty $P(\pi)$ for each tested road network when comparing our cooperative planning algorithms to the baseline non-cooperative planner. The proposed cooperative algorithms brings us significantly closer to the ideal penalty of 0, which is nearly unattainable in most cases. Achieving a 0 penalty would require each EV to take the geographically shortest path without any waiting time, implying that the baseline planner already provided the optimal solution — a rare occurrence in practice. The table demonstrates that each proposed cooperative planner reduces the global penalty of the solution by over 90%. For instance, in the Québec_{347} road network with 128 EVs, the penalty decreased by 24 000 minutes. Therefore, on average, each EV's trip was shortened by 187 minutes.

6 CONCLUSION

The main contributions of this paper are two-fold. First, we introduced the Cooperative Electric Vehicles Planning Problem (CEVPP) and demonstrated through experiments that incorporating cooperative algorithms can significantly reduce overall time, highlighting the real-world importance of this issue. Second, we introduced both an optimal and three variants of an approximate cooperative planner, which we evaluated using three real-world road networks. Our most effective cooperative planner reduced the mean total trip time of each EV driver by 2 hours. As the number of EVs on the road network grows, the number of bottlenecks at charging stations that

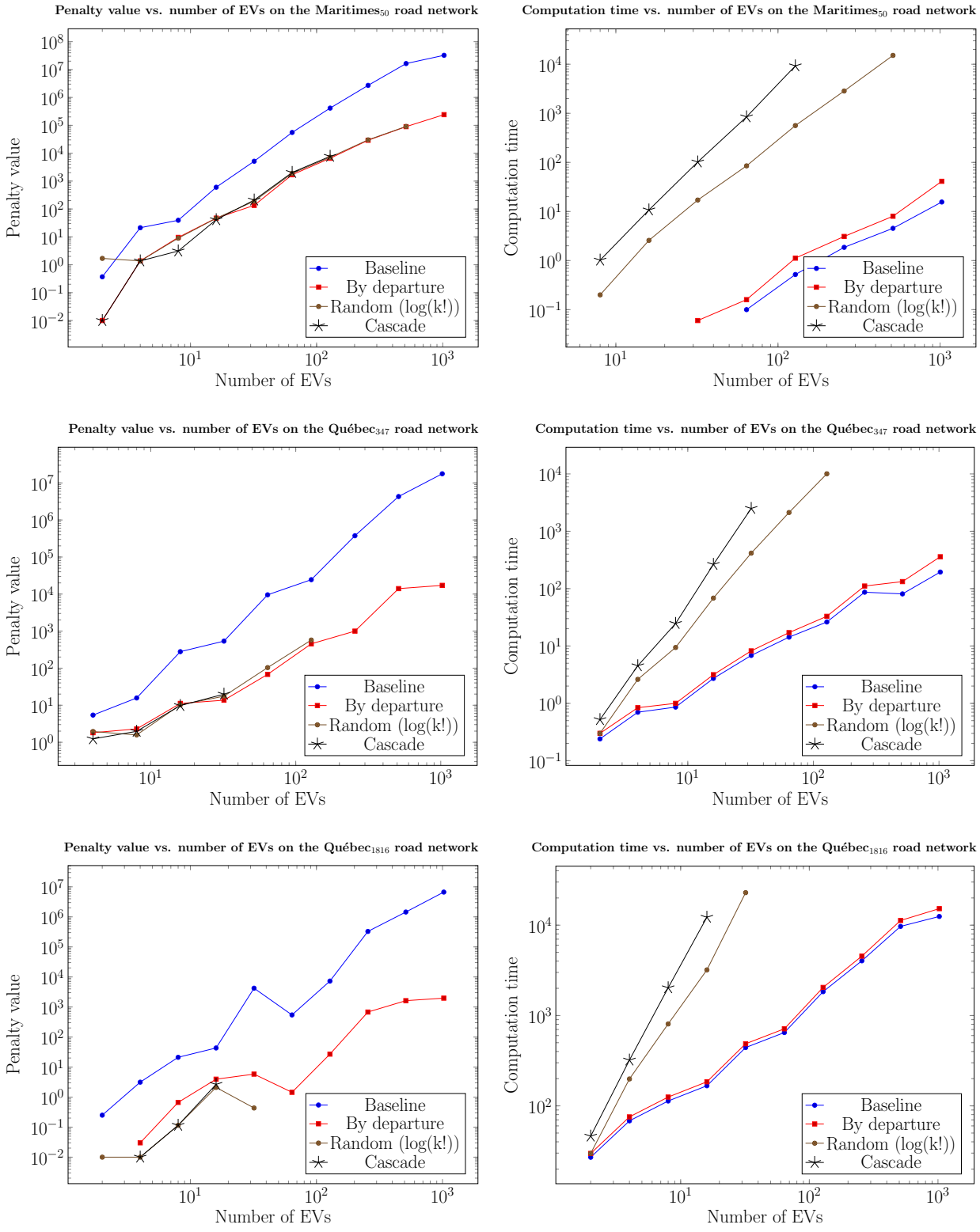

Figure 2: Penalty $P(\pi)$ and running times (in s) of the competing algorithms for the three considered road networks (Maritimes$_{50}$, Québec$_{347}$ and Québec$_{1816}$) when varying the number of EVs from 2 to 1024.

Table 1: Results (P: penalty $P(\pi)$, and RT: running time) of the compared planners, the Baseline non-cooperative and the permutations (ordered by departure, random $\log(k!)$, and cascade variants) cooperative planner, on three different road networks. For each test, we report the average of 50 CEVPP requests. The symbol '-' indicates that the algorithm did not manage to find a solution for the batch of 50 requests within 15 minutes. The lowest penalty value on each row is bolded.

Test characteristics		Baseline		Ordered by departure		Random $\log(k!)$		Cascade	
Network	**#EVs**	**P**(π) (min)	**RT** (ms)	**P**(π) (min)	**RT** (ms)	**P**(π) (min)	**RT** (ms)	**P**(π) (min)	**RT** (ms)
Maritimes_{50}	2	0.37	0	**0.01**	0	**0.01**	0	**0.01**	0
Maritimes_{50}	4	21.275	0	**1.37**	0	**1.37**	0	**1.37**	0
Maritimes_{50}	8	39.355	0	9.58	0	8.98	0.2	**3.11**	1.02
Maritimes_{50}	16	602.378	0	46.9963	0	46.4562	2.58	**41.1537**	10.74
Maritimes_{50}	32	5151.81	0	**134.741**	0.06	189.81	17.1	208.724	101.92
Maritimes_{50}	64	55471.1	0.1	**1687.7**	0.16	1893.06	85.04	2038.08	847.16
Maritimes_{50}	128	414601	0.52	**6589.36**	1.12	7183.98	562.5	7725.53	9253.56
Maritimes_{50}	256	2706313	1.86	**29106.8**	3.1	30094.6	2850.12	-	-
Maritimes_{50}	512	16389675	4.54	**89585.1**	8	90916.9	15135.6	-	-
Maritimes_{50}	1024	32440149	15.6	**241419**	41.22	-	-	-	-
Québec_{347}	2	**0**	0.24	**0**	0.3	**0**	0.3	**0**	0.52
Québec_{347}	4	5.415	0.7	1.815	0.84	1.955	2.62	**1.235**	4.52
Québec_{347}	8	15.695	0.86	2.2925	1	**1.5775**	9.42	1.975	24.68
Québec_{347}	16	279.127	2.72	11.1913	3.16	10.2212	68.36	**9.78751**	264.38
Québec_{347}	32	537.111	6.84	**13.6631**	8.22	17.5406	415.64	19.8519	2500.22
Québec_{347}	64	9549.65	14.26	**67.9469**	17.1	103.703	2123.74	-	-
Québec_{347}	128	24454.8	26.2	**454.282**	32.92	573.711	10046.9	-	-
Québec_{347}	256	375362	86.64	**1000.89**	111.04	-	-	-	-
Québec_{347}	512	4288950	80.94	**14037**	132.6	-	-	-	-
Québec_{347}	1024	17657016	194.54	**17172.4**	359.92	-	-	-	-
Québec_{1816}	2	0.25	26.98	**0**	29.74	0.01	29.8	**0**	46.22
Québec_{1816}	4	3.14	68.26	0.03	75.52	**0.01**	198.52	**0.01**	321.02
Québec_{1816}	8	21.1175	113.2	0.665	125.14	**0.115**	806.82	**0.115**	2027.16
Québec_{1816}	16	43.1937	166.92	3.94625	184.64	**2.10125**	3197.58	2.65	12232
Québec_{1816}	32	4219.4	442	5.81813	486.88	**0.43375**	22914.8	-	-
Québec_{1816}	64	543.543	647.94	**1.44281**	712.74	-	-	-	-
Québec_{1816}	128	7294.58	1835.3	**26.9613**	2045.88	-	-	-	-
Québec_{1816}	256	327805	4031.76	**678.371**	4540.98	-	-	-	-
Québec_{1816}	512	1451887	9694.42	**1623.29**	11267.2	-	-	-	-
Québec_{1816}	1024	6695855	12483.7	**1970.67**	15233.8	-	-	-	-

Table 2: Average percentage reduction in penalty $P(\pi)$ for each tested road network between the compared planners.

Network	**By departure**	**Random** $\log(k!)$	**Cascade**
Maritimes_{50}	93.06	93.07	95.22
Quebec_{347}	86.33	86.73	89.35
Quebec_{1816}	96.69	97.57	98.25
Average	92.03	92.46	94.27

lead to waiting times will increase, presenting more opportunities for optimization and further emphasizing the relevance of CEVPP.

As future work, we aim to explore methods for pruning substantial portions of the state space utilized by our optimal cooperative planner, potentially drawing inspiration from the M* algorithm. This would enable a significant reduction in spatial and temporal resources, making the planner more practical for real-world applications. Additionally, we plan to conduct a comprehensive analysis of various permutation subsets for our permutations cooperative planner. Lastly, we intend to incorporate more factors into CEVPP, such as accounting for waiting times at charging stations caused by external EVs, e.g., by taking account of historical charging stations occupancy or by considering a reservation system that each EV must use, to further enhance its applicability and effectiveness.

ACKNOWLEDGMENTS

This research has been supported by the *Natural Sciences and Engineering Research Council of Canada* (NSERC) and the *Fonds de Recherche du Québec — Nature et Technologies* (FRQNT).

REFERENCES

[1] Saeed Nasehi Basharzad, Farhana M. Choudhury, Egemen Tanin, Lachlan L. H. Andrew, Hanan Samet, and Majid Sarvi. 2022. Electric Vehicle Charging: It Is Not as Simple as Charging a Smartphone (Vision Paper). In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*. ACM, Seattle Washington, 1–4. https://doi.org/10.1145/3557915.3560967

[2] Moritz Baum, Julian Dibbelt, Andreas Gemsa, Dorothea Wagner, and Tobias Zündorf. 2015. Shortest Feasible Paths with Charging Stops for Battery Electric Vehicles. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL '15)*. Association for Computing Machinery, New York, NY, USA, 1–10. https://doi.org/10.1145/2820783.2820826

[3] Yue Cao, Tong Wang, Omprakash Kaiwartya, Geyong Min, Naveed Ahmad, and Abdul Hanan Abdullah. 2018. An EV Charging Management System Concerning Drivers' Trip Duration and Mobility Uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48, 4 (2018), 596–607. https://doi.org/10.1109/TSMC.2016.2613600

[4] Jaël Champagne Gareau, Éric Beaudry, and Vladimir Makarenkov. 2019. 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, 389–397. https://doi.org/10.1145/3347146.3359064

[5] Tomislav Erdelić and Tonči Carić. 2019. A Survey on the Electric Vehicle Routing Problem: Variants and Solution Approaches. *Journal of Advanced Transportation* 2019 (2019), 1–48. https://doi.org/10.1155/2019/5075671

[6] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. 2012. Exact Routing in Large Road Networks Using Contraction Hierarchies. *Transportation Science* 46, 3 (2012), 388–404. https://doi.org/10.1287/trsc.1110.0401

[7] Peter Hart, Nils Nilsson, and Raphael Bertram. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans - TSMCA* , 4, 2 (1968), 100–107. https://doi.org/10.1109/tssc.1968.300136

[8] Christopher Hecht, Karoline Victor, Sebastian Zurmühlen, and Dirk Uwe Sauer. 2021. Electric Vehicle Route Planning Using Real-World Charging Infrastructure in Germany. *eTransportation* 10 (2021), 100143. https://doi.org/10.1016/j.etran.2021.100143

[9] Hydro-Québec. 2023. The Electric Circuit. Retrieved 2023-11-12 from https://lecircuitelectrique.com/en/

[10] Nicholas Kullman, Justin Goodson, and Jorge E Mendoza. 2017. Electric Vehicle Routing with Uncertain Charging Station Availability Dynamic Decision Making. In *INFORMS Transportation and Logistics Society Triennial Conference*. Chicago. https://doi.org/hal-01497793

[11] Jean-Claude Latombe. 2012. *Robot Motion Planning*. Springer Science & Business Media.

[12] Erwin Lejeune and Sampreet Sarkar. 2021. A Survey of the Multi-Agent Pathfinding Problem. https://doi.org/10.13140/RG.2.2.14030.28486

[13] Jane Lin, Wei Zhou, and Ouri Wolfson. 2016. Electric Vehicle Routing Problem. In *Transportation Research Procedia*, Vol. 12. Elsevier, 508–521. https://doi.org/10.1016/j.trpro.2016.02.007

[14] Alexandre Lucas, Ricardo Barranco, and Nazir Refa. 2019. EV Idle Time Estimation on Charging Infrastructure, Comparing Supervised Machine Learning Regressions. *Energies* 12, 2 (2019), 269. https://doi.org/10.3390/en12020269

[15] Prashanth Rajagopalan, Jack Thornby, and Prakash Ranganathan. 2023. Short-Term Electric Vehicle Demand Forecasts and Vehicle-to-Grid (V2G) Idle- Time Estimation Using Machine Learning. In *2023 IEEE 13th Annual Computing and Communication Workshop and Conference (CCWC)*. 1279–1286. https://doi.org/10.1109/CCWC57344.2023.10099356

[16] Martin Sachenbacher, Martin Leucker, Andreas Artmeier, and Julian Haselmayr. 2011. Efficient Energy-Optimal Routing for Electric Vehicles. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*. AAAI Press, 1402–1407.

[17] David Silver. 2005. Cooperative Pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 1, 1 (2005), 117–122. https://doi.org/10.1609/aiide.v1i1.18726

[18] Roni Stern. 2019. Multi-Agent Path Finding – An Overview. In *Artificial Intelligence*, Gennady S. Osipov, Aleksandr I. Panov, and Konstantin S. Yakovlev (Eds.). Vol. 11866. Springer International Publishing, Cham, 96–115. https://doi.org/10.1007/978-3-030-33274-7_6

[19] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Kumar, Roman Barták, and Eli Boyarski. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Proceedings of the International Symposium on Combinatorial Search* 10, 1 (2019), 151–158. https://doi.org/10.1609/socs.v10i1.18510

[20] Timothy M Sweda, Irina S Dolinskaya, and Diego Klabjan. 2017. Adaptive Routing and Recharging Policies for Electric Vehicles. *Transportation Science* 51, 4 (2017), 1326–1348. https://doi.org/10.1287/trsc.2016.0724

[21] Glenn Wagner and Howie Choset. 2011. M*: A Complete Multirobot Path Planning Algorithm with Performance Bounds. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 3260–3267. https://doi.org/10.1109/IROS.2011.6095022

[22] Patrick Weber and Muki Haklay. 2008. OpenStreetMap: user-generated street maps. *IEEE Pervasive Computing* 7, 4 (2008), 12–18. www.openstreetmap.org

[23] A. Zelinsky. 1992. A Mobile Robot Exploration Algorithm. *IEEE Transactions on Robotics and Automation* 8, 6 (1992), 707–717. https://doi.org/10.1109/70.182671