Optimization of Routes of Electric Vehicles taking into account the Status of Charging Stations

Gonçalo Cravino Fernandes
Instituto Superior Técnico - Universidade de Lisboa
Lisboa, Portugal
goncalo.c.fernandes@tecnico.ulisboa.pt

***Abstract*—In the last decade, the rise of electric vehicles(EV) has brought benefits such as reduced reliance on fossil fuels. however, the limited-range batteries and charging challenges have created the need for advanced electric vehicle route planning. This thesis explores three different methodologies to solve the electric vehicle route planning problem, minimizing both route duration and cost. The three methodologies consist of, a search-based approach utilizing the Dijkstra's algorithm, to obtain the optimal path for our problem formulation. The second methodology, consists of a Meta-heuristic approach utilizing a Genetic algorithm, to obtain a close-to-optimal solution in a faster runtime. The third and final methodology, consists of a reinforcement learning approach utilizing the Deep Q-learning algorithm, to obtain a close-to-optimal solution in real time. The experimental results obtained showed the capability of the Genetic algorithm to obtain solutions comparable quality to those achieved by Dijkstra's algorithm, but with significantly improved runtime efficiency. The results obtained from the Deep Q-learning algorithm, although underwhelming showed the potential of the algorithm utilization.**

***Index Terms*—Electric vehicle route planing, Dijkstra, Genetic algorithm, Deep Q-Learning**

I. Introduction

A. Context and Motivation

In the last decade, the utilization of EVs has increased significantly [1]. With all the advantages that EVs have brought especially in terms of their independence from fossil fuel, their limited-range batteries have created a new constraint in Eletric vehicle route planing(EVRP). EV's battery sizes have been continuously increasing [2] to the point where they are now capable of making long-distance routes without the constant need for charging [1].

However, due to the time associated with the charging of EVs, users tend to be overcautious about their EV range, especially when performing long-distance routes. This compounded with the possible waiting times due to the lack of infrastructure [3] shows the need for solutions for EVRP problems. In addition to waiting times at charging stations (CS) due to rising electricity prices, EV users are increasingly more interested in minimizing energy consumption and charging costs [4]. It has been concluded that, in comparison with conventional route planning, EVRP is more challenging mainly due to the time variables associated with EV charging such as the state of charge, the power of the charging station and the occupancy state of the charging station [5], [6]. This routing problem also becomes more complex when taking into consideration the different charging costs of charging stations and the possibility of prioritizing charging costs and energy consumption over route duration.

With this context in mind, the focus of this work is to explore the different algorithms capable of finding the optimal route from a starting location to a goal location in a manner that optimizes both time and cost. Our specific goal is, to find the necessary charging stations in which to make a charging operation, in order to achieve our final location. To find the correct charging stations, our algorithm needs to be able to select the best charging stations to achieve our cost and time minimization goal, on the characteristics of the charging stations such as location, charging cost, and charging power. By leveraging these station characteristics, the algorithm should be able to find the route that achieves the best balance between route duration and route cost.

B. Background

1) Dijkstra's Algorithm: Dijkstra's algorithm is one of the widely used algorithms to solve SPSP and SSSP in a directed graph with only positive edge weights. The algorithm was first published by Edsger Dijkstra in 1959 [7]. The algorithm receives as input a directed graph, an edge weight function $\omega \to R_0^+$, and a source vertex. The output of the algorithm is the shortest path tree from the source vertex to all the other vertices in the graph, with this shortest path tree we are able to obtain the shortest path and distance value from the source to any given vertex of the graph.

2) Meta-Heuristic Algorithms: Meta-heuristic algorithms are a set of problem-solving algorithms that unlike exact algorithms instead of finding the optimal solution no matter the time and computational cost this algorithm focus on obtaining a close to optimal solution at a better time and computational cost. These algorithms are very useful in applications where obtaining a solution quickly is more valuable than obtaining the optimal solution.

3) Genetic Algorithms: Genetic algorithms are a problem-solving algorithm that works based on the idea of natural selection and evolution. The basis of the algorithm is to create a population of possible solutions and assign them a given fitness value depending on their performance for the given problem. By assigning a greater likelihood of being used to generate a new generation to the best-fitting solutions,

the algorithm evolves to create better solutions with each generation.

The Genetic algorithm is characterized by representing the problem solution as a chromosome where parts of the solution can be separated into genes in order to apply mutation and crossover mechanisms to these chromosomes.

These mutation and crossover mechanisms are the key mechanisms that prevent the algorithm from converging in local minimums and allowing close to optimal solutions to be found. The crossover mechanism is the mechanism that combines the chromosomes from two different solutions in order to create a solution with the best parts of both solutions. The mutation mechanism is the mechanism that allows the introduction of new solution paths to the algorithm by changing part of a solution´s chromosome with new genes.

4) Markov Decision Process: Markov Decision Process (MDP) is a mathematical framework for modeling decision-making. A Markov decision process is defined by a set of states and actions. These actions allow for the transitions between states and the outcome of these transitions allows us to evaluate these state-action pairs. In an MDP an agent in a given state at a given time as a set of actions that can be performed based on its current state and goal. By taking an action the state of this agent changes to a new one, a reward is given to the agent based on this action which could be positive or negative. The goal in an MDP is to find a policy, which is a set of rules that the agent can follow to maximize the expected reward over time.

5) Reinforcement Learning: Reinforcement learning is a field of machine learning that focuses on training an agent by rewarding (and punishing) the agent depending on the actions taken, in other words, is learning to act by trial and error. In reinforcement learning, every time an action is performed by an agent in a given state reward is given to the agent and the goal of the agent is to maximize the sum of the reward.

Reinforcement learning in contrast with supervised learning does not need labeled training to be trained to learn the correct behavior but instead the training process rewards the model when it acts as intended and punishes it otherwise. In this aspect, reinforcement learning is more similar to biological learning where learning comes from rewards and punishments from trial and error.

Reinforcement learning algorithms can be divided into two categories, model-based, and model-free algorithms. In model-based algorithms as the name suggests the agent models its environment and from this model predicts future rewards without needing to perform those future actions. Model-free algorithms, in contrast, do not need a model of the environment of the agent and can only know the reward of a given action by performing it, this means that model-free algorithms will repeat the same actions multiple times updating their action-taking strategy based on the outcome of their actions, to maximize rewards.

For our approach, we are going to focus on the Q-learning and deep Q-learning algorithms that due to it being a model free Reinforcement Learning (RL) algorithm which is necessary for a complex environment like ours where is difficult to obtain a reliable model, and due to its trial and error learning approach that helps as lead with complexity and unpredictability of the interaction of our agent and its environment.

6) Q-Learning: Q-learning (QL) [8] is a model-free, policy-free reinforcement learning algorithm. This algorithm aims to find the best action the agent can perform for a given state. the Q-learning algorithm is considered off-policy for being able to learn from taking random actions, making the need for a policy not necessary. The way the algorithm works is by having a quality function for every state-action pair the agent can perform, by exploring the environment the model updates this quality function with the reward obtained from each action performed by the agent for a given state. Given enough training, the Q-learning algorithm is sure to converge to an approximation of the best action-value function for a given goal.

7) Deep Q-Learning: Deep Q-learning (DQL) is a deep reinforcement learning algorithm that picks up on the concepts of Q-learning but uses a deep neural network to approximate the Q-function. In traditional Q-learning, the Q-function is typically represented as a table of state-action values. This can be impractical for large or continuous state spaces because the size of the table grows exponentially with the number of states and actions. Deep Q-learning addresses this problem by using a neural network to approximate the Q-function. The neural network takes as input the current state of the environment and outputs a predicted Q-value for each possible action. The Q-values are then used to choose the next action in the same way as in traditional Q-learning.

Due to its complexity EVRP is impractical to solve with a look-up table of state-action and therefore the use of neural networks to compute the state-action values is necessary. In [9], [10], [11] we see the use of neural networks to improve the Q-learning algorithm with positive results

One of the key challenges in using deep Q-learning is that the Q-function is not differentiable, which means that standard backpropagation cannot be used to train the neural network. Instead, the network is trained using a variant of stochastic gradient descent called the Q-learning update rule. This update rule adjusts the network weights to minimize the error between the predicted Q-values and the target Q-values, which are the expected rewards of taking each action in the current state.

8) DNDQL: The Dueling Network Deep Q-Learning(DNDQL) algorithm [12] is a variation of the DQL algorithm that was created to improve the efficiency and effectiveness of deep Q-learning, particularly in situations where it is challenging to estimate the values of different

actions accurately. The main difference between DQL and DNDQL is the Q-value estimation. While DQL uses a neural network to estimate its Q-values, the DNDQL decomposes the Q-value of a state-action pair into two components:

- State Value: which estimates the value of being in a particular given state.
- Action Advantage Value: which estimates the advantage of taking each possible action in the given state, capturing the difference in expected rewards between different actions in the same state.

This decomposition improves the algorithm's learning efficiency and can lead to more stable and faster convergence for reinforcement learning tasks in complex environments.

C. State of the Art

1) Conventional Search-based Approach to EVRP : The traditional approach to Electric Vehicle Route Planning (EVRP) involves using graph search algorithms like Dijkstra, Bellman-Ford, and heuristic accelerated algorithms such as A-star. These algorithms can be adapted for EVRP by creating a label set for each vertex, enabling them to handle multivariable problems. Some adaptations, like the one in Liu et al. [6], optimize both travel time and energy recharging costs. Heuristic accelerated algorithms, as seen in Schoenberg et al. [5] and Zhang [13] enhance runtime efficiency compared to standard search-based methods. To apply these algorithms effectively, the road network is transformed into a graph network with charging stations as nodes. Preprocessing is required to select the most relevant part of the road network, improving algorithm runtime and reducing system complexity. While this preprocessing works well for short routes in small road networks, it becomes less suitable for long-distance routes due to increased graph complexity and runtime limitations tied to the graph's size.

2) Meta-Heuristic Algorithms in EVRP: In the context of Electric Vehicle Routing Problems (EVRP), classical shortest path algorithms like Dijkstra face limitations due to exponential time complexity, especially in large graphs. To address this challenge, meta-heuristic algorithms like Genetic Algorithms (GA) have been employed, offering faster but near-optimal solutions. In Shao et al. [14], a GA was used to optimize routes for electric delivery vehicles, minimizing monetary costs in a realistic road network. The study found that GA achieved acceptable performance in terms of computational time, convergence, and solution quality. In Li et al. [15], a variant of GA, the re-insertion Genetic Algorithm, was used to optimize routes for electric delivery vehicles, achieving better convergence and performance by focusing on correlated stops. Barco et al.[16], employed Differential Evolution (DE), an evolutionary algorithm, to optimize routes and charge schedules for electric transport shuttles, demonstrating faster convergence with performance close to GA. These approaches show the effectiveness of meta-heuristic algorithms in tackling EVRP challenges.

3) Reinforcement Learning in EVRP: Reinforcement learning algorithms have emerged as a promising solution for Electric Vehicle Routing Problems (EVRP) to address limitations in classical shortest path algorithms and meta-heuristic-based methods. In Lee et al. [9], Deep Q-Learning (DQL) is employed to optimize routes for multiple EVs to charging stations, reducing waiting times and traffic in a known network. In Qian et al. [10], a similar approach focuses on optimizing the route of a single EV to charging stations, utilizing feature extraction to adapt to varying traffic conditions and energy prices, effectively minimizing recharging costs. Zhang [11] introduces the use of a dueling deep Q-learning algorithm to optimize EV routes, particularly suitable for EVRP. "Lee et al. [9] and "Qian et al. [10] demonstrate successful applications in small local networks but are limited in handling long-distance routes with multiple recharges, while "Qian et al. [10] excels in recharging cost minimization. Combining the strengths of these approaches with the methods in Zhang [11] holds the potential for further route optimization in diverse EVRP scenarios.

D. Paper Organization

The paper is structured as follows: after this introduction, it is presented the formulation EVRP problem and the formulation of our environment. The methodologies developed using the Dijkstra's algorithm , the Genetic algorithm and the Deep Q-learning algorithm to solve our formulation are presented in section II. Section III presents the results obtained from the three different methodologies. Section IV assembles the main conclusions.

II. Methodology

A. Problem Formulation

In the Section 1a is presented the Formulation for our route and charging optimization problem in a mathematical formulation based in the formulation for electric vehicle route and planning formulated in [17], [10], [11], where all the variables are defined in Table I.

TABLE I
FORMULATION VARIABLES.

Symbol	Description	unit
V	Set of nodes	
s	Starting node	
g	Goal node	
(i,j)	Path between node i and node j	
td_{ij}	Drive time between node i and node j	hour
tc_j	Charging time of charging operation in node j	hour
cc_j	Charging cost of charging operation in node j	Euro €
x_{ij}	Binary decision variable to identify selected route	0-1
y_j	Binary decision variable to identify selected charging station	0-1
γ	Multi-objective optimization weight	

$$\text{minimize} \quad \sum_{i,j \in E, i \neq j} \Big[(cc_j y_j)\gamma + (td_{ij} x_{ij} + tc_j y_j)(1-\gamma) \Big] \tag{1a}$$

$$\text{subject to} \quad \sum x_{sj} - \sum x_{js} = 1 \qquad \forall j \in V, \tag{1b}$$

$$\sum x_{gi} - \sum x_{ig} = -1 \qquad \forall i \in V, \tag{1c}$$

$$x_{ij} - y_j \geq 0 \qquad \forall (i,j) \in V, \tag{1d}$$

$$x_{ij} = \{0,1\} \qquad \forall (i,j) \in V, \tag{1e}$$

$$y_j = \{0,1\} \qquad \forall j \in V \tag{1f}$$

Constraints 1b and 1c ensures the restriction imposed by the EVRP, where the EV always starts at a vertex s and always arrives at a vertex g by following a sequential path of vertices $\in V$. Constraint 1d ensures the EV is only able to receive energy from charging stations that are in a reachable path. Constraint 1e defines the binary decision variable x_{ij} which indicates whether the EV has followed the path from i to j or not. Constraint 1f defines the binary decision variable y_j which when $y_j = 1$ indicates that the path from i to j leads to a charging station where the EV has received energy and when $y_{ij} = 0$ indicates the path from i to j does not lead to a charging station and the EV has not received energy.

B. Environment Formulation

In order to implement our solutions to the EVRP problem, we need to be able to model not only the real-life road network and charging infrastructure but also the behavior of a typical EV.

1) Routing API: In order to verify if the chosen charging stations are the best, we need to estimate the best route between them, and correctly estimate the time and energy required to perform these routes. For our approach, we relied on an open-source software called OpenRouteService(ORS) [18], which provides multiple services based on crowd-sourced geographical data from OpenStreetMap [19]. For our solution, we used a local client of their software which allows us to execute unlimited requests of routes, and used their route service by making a query with an initial coordinate and final coordinate to the service. This query provided us with an estimation of the time required to perform the route, the road distance in km of the route, and an array of coordinates detailing the said route.

2) Charging Station Pre-processing: The basis of EVRP settles in deciding which charging station a given EV should stop to refuel. Therefore the model of this charging infrastructure needs to be as close as possible to reality. In order to achieve this we relied upon the acquisition of data from an open-source API called OpenChargeMap(OCM) [20]. With this API we were able to make a query to obtain the data from all charging stations in their database for a given geographic area. The data provided by this API ranged from its geographic coordinates to the number and power of the charging station ports.

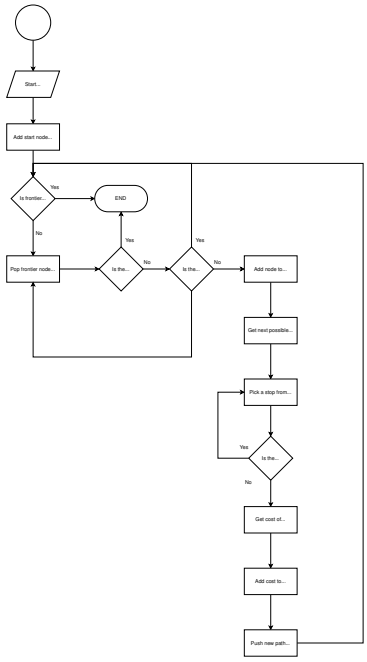

Fig. 1. Implemented Dijkstra's algorithm diagram.

C. EVRP usign Dijkstra

In order to solve the problem detailed in Section II-A, we will employ the use of the Dijkstra's algorithm to find the optimal route for our formulation. Dijkstra's algorithm is a search-based algorithm used to find the shortest path between nodes in a given graph. By systematically exploring and evaluating potential routes, it enables us to determine the most efficient solution to our problem.

In figure 1 is depicted the flowchart design of our implementation of the Dijkstra algorithm to solve our EVRP problem.

1) EVRP Graph Formulation: In our environment, we can see that the road infrastructure and the charging stations already behave similarly to a graph where the roads connect the multiple charging stations just like edges connect multiple nodes in a graph, therefore, the most logical approach is to use each charging station as the graph node and the graph edges represent the cost associated to moving from one station to another.

Nodes

Each charging station has different characteristics, these characteristics are what make a charging station a good or a poor choice for recharging. Therefore these parameters need to be well described in their corresponding node. For this, we described each node with a dictionary containing the most important of these characteristics, this being its geographic

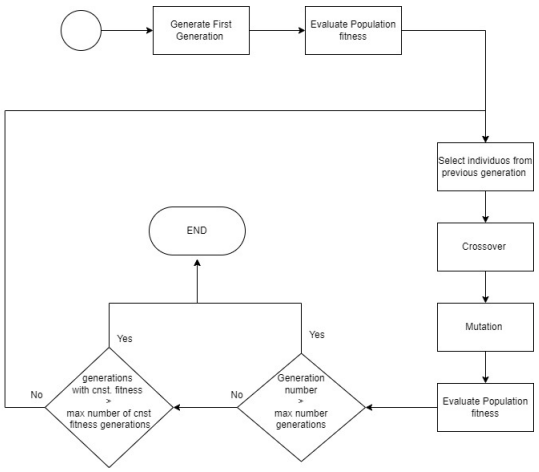

Fig. 2. Implemented GA diagram.

coordinates in order to calculate the route characteristics to reach this CS, and the number of ports and their respective characteristics in order to calculate the charging characteristics of the CS.

Edges

In a weighted graph, the edges of the graph describe the cost associated with moving from one particular node to another. In our environment, this cost refers not only to the time cost associated with driving from one node to another but also the time cost associated with the charging operation performed at the new node and its associated monetary cost.

States

In our algorithm, we want to find the path with the lowest cost from the initial state to the goal state. We defined a state as a tuple containing the current node, the EV battery state percentage, and the path done to achieve the current node.

D. EVRP using Genetic Algorithm

In order to solve our problem formulation defined in Section II-A, we are going to employ the implementation of a Genetic algorithm. Genetic Algorithms operate by utilizing genetic mechanisms that mimic the principles of natural selection, crossover, and mutation to iteratively enhance solution populations across generations. This approach enables us to systematically refine and evolve solutions to the specified problem.

In Figure 2 is depicted the flowchart design of our implementation of the GA to solve our EVRP problem

1) Chromosome: The first step in any GA implementation is to define our chromosomes. This chromosome describes the behavior of each individual and is in this chromosome that the crossover and mutation mechanisms will make changes.

In our implementation, the chromosome consisted of a list where the first element contains the number of stops of the individual. The second element consists of a list of the ordered stop nodes that define the individual's route. The last element consists of a list with the order index of the stop nodes of the individual's route.

In order for the algorithm to function as intended only valid chromosomes were accepted. For a chromosome to be deemed valid it must satisfy these four rules:

1) The number of stops must be between the maximum and minimum number of stops for a given route.
2) Each stop must be reachable from the previous stops.
3) The first stop must be reachable from the starting position.
4) The target position must be reachable from the last stop.

2) First Generation Creation: To create viable individuals, the algorithm starts by defining for the given start and target position the list of stops reachable from the start position, the list of stops that can reach the target position, the minimum number of stops, and the maximum number of stops. The minimum number of stops was calculated by getting the road distance between the initial position and the final position and then dividing this distance by the EV driving range. The maximum number of stops is then defined as $max\_N\_stops = 2 \times min\_N\_stops + 1$.

With these lists and values, to create each viable individual the algorithm first chooses the number of stops of the individual from a random integer between the minimum number of stops and the max number of stops. After that the first stop is randomly selected from the list of stops reachable from the start position, this ensures the 3rd rule is always met. The subsequent stops until the last stop are randomly selected from the stops that are not reachable from the start position and do not reach the goal position. finally, the last stop is randomly selected from the list of stops that reach the goal position also ensuring the following of the 4th rule. After the initial creation of the individual in order to ensure that the 2nd rule is followed the individual is evaluated to check if all stops are reachable from the previous stop, if during the evaluation the algorithm detects that one stop is not reachable from the previous stop a new stop is randomly selected until a reachable stop is found.

3) Fitness Function: The objective of the GA is to find the solutions most suited to solve our problem. In order to find this solution there needs to be defined a fitness parameter to evaluate how good the solution of a given individual of the population is. This fitness parameter needs to correctly describe the performance of the individual in solving the problem at hand. For our problem, the fitness value was computed from two components:

- The time component (C_{time}) - consisting of the time associated with driving between stations and the time associated with the recharging operations.
- The cost component (C_{cost}) - consisting of the cost associated with the recharging operation.

The time component is defined in Equation 2 by adding the total driving time of the chromosomes route with the total charging time. In order to penalize unnecessary stops, an extra term is added to represent the time cost of stopping in a charging station. This cost represents the real-time cost of arriving at the charging station, the time of connecting the car to the charging station, and the time of disconnecting and paying for the charging operation. This charging station cost is constant for every station.

$$C_{time} = \sum t_{driving} + \sum t_{charging} + \sum t_{CS} \tag{2}$$

The cost component is only comprised of the sum of the cost of all of the charging operations of the route and is defined in Equation 3.

$$C_{cost} = \sum m_{charging} \tag{3}$$

In order to combine these components in a single cost, a weight γ is used to find an optimization balance between the two components. this is combined cost is defined in Equation 4.

$$C_{combined} = (C_{cost})\gamma + (C_{time})(1-\gamma) \tag{4}$$

This combined cost, however, increases as the time component and the cost component of the route increases. This would mean that the fittest individuals would be the ones that took the most time and cost consuming route. Therefore, for the fittest individuals to be the most time and cost saving routes, our fitness score consists of the inverse value of the combined cost. This is defined in Equation 5.

$$Fitness = \frac{1}{C_{combined}} \tag{5}$$

4) Selection Mechanism: To create a new generation a method to select individuals is necessary not only to select good individuals for our new generations but also to select good individuals to use in our crossover and mutation mechanisms. In order to achieve this selection method we utilized two common selection algorithms:

- The tournament selector.
- The biased roulette selector.

In order to get the benefits of both methods, whenever a selection of an individual of the population was needed, one of these methods was chosen randomly with the same probability for either one.

5) Crossover Mechanism: The crossover mechanism is an important part of the generation creation.

The basis for the crossover mechanism is to select two individuals and combine parts of their chromosomes to create a new individual chromosome. In our problem, the sequence of the stops is as important as the stops themselves. Therefore the combination of the two chromosomes needs to preserve good sequences of stops.

To achieve this, in our implementation, the algorithm selects two individuals from the old generation to be our parent individuals. Then the number of stops of the new "child" individual is chosen to be the number of stops of one of the parent individuals.

After this, in order to preserve sequences of stops, the algorithm splits the child's chromosome into a randomly chosen point, between the first stop and the last stop. This split creates two chromosome segments, the first segment from the 1st stop to the splitting point, and the second segment from the splitting point to the final stop.

In order to create this new chromosome consisting of two chromosome segments, we create 4 chromosome segments from the parent chromosome. For the first chromosome segment of the child, for a given number of stops N, we use the first N stops of each parent to create two chromosome segments. For the second chromosome segment of the child, for a given number of stops X, we use the final X stops of each parent to create the other two chromosome segments.

To find the best combination of the parent chromosomes, two child chromosomes are created. In one of the child chromosomes, the first segment comes from parent A and the second segment from parent B and the reverse for the other child. Then each child is evaluated and the best child is the one chosen for the new generation.

6) Mutation Mechanism: The mutation mechanism is the mechanism that brings genetic variability to the solution population. By having a mutation mechanism we allow the introduction of new chromosome combinations that haven't yet been seen in the population. This allows us to achieve a solution that is closer to the optimal solution without depending on the initial population chromosomes.

For our implementation, we consider a mutation as changing one of the stops of the chromosome with a new stop. Due to the nature of our problem, swapping one stop for another random stop would in the majority of cases, lead to an invalid chromosome due to having an unreachable stop. To prevent this, instead of choosing a random stop from the entire stop list, we only consider stops that are within a 20 km radius of the stop we want to replace. Depending on the number of stops of the chromosome we want to apply the mutation to, a number of mutations are randomly selected between 1 mutation and $\frac{N\_stops}{2}$. The algorithm then randomly selects which of the stops of the chromosome suffers a mutation.

7) Fixing Mechanism: During The creation of the new generation, as a result of crossover and mutation, these mechanisms only focused on checking the validity of these chromosomes, but some of these chromosomes might contain redundant segments. For example, we can have a chromosome that has 5 stops but stop 5 is reachable from stop 2 making

stops 3 and 4 unnecessary. In order to verify this every new candidate for the new generation is tested for unnecessary stops before being added to the new generation.

To achieve this an algorithm was developed that checked for every stop i where $i \neq$ final stop if existed any stop j where $j \geq i+1$ that can be reached from i and maintains the chromosome viable.

This fixing mechanism helps to accelerate the algorithm by removing viable but low-fitness solutions from being added to the new generations.

E. EVRP using Deep Q-learning

In order to solve the our EVRP problem we used a RL based approach. This approach is based on the DQL algorithm that use a deep neural network to approximate the value-action function.

1) OpenAI Gym Environment: In order to implement the DQL algorithm in our problem, we developed a custom gym environment using the openAI gym RL API.

To create a custom gym environment we had to define:

- The action space
- The observation space
- the reset and step function
- the reward function

Action Space

In our problem, we define an action as choosing which station to travel to from the current state, therefore our action space is comprised of the total possible stations the agent can choose from.

Observation Space

The observation space is comprised of all the variables important in our environment. In our implementation, we used a Python dictionary to define our observation. This dictionary was comprised of four keys:

- 'agent' - containing the agent's current station index, latitude, longitude, and battery state
- 'path' - listing all the already visited stops.
- 'start' - containing the start geographic coordinates.
- 'goal' - containing the goal geographic coordinates.
- 'stops' - consisting of an array containing all station indexes, geographic coordinates, charging power, and charging cost.

Reset and Step function

For the correct function of the gym environment, we need to develop a reset function that assigns the initial state values for each variable of the system. Ans a step function that for the given action chosen, computes its reward value updates the observation, and checks for termination of the episode.

2) Reward Funtion: The reward function determines the behavior our agent takes.For our problem, the intended behavior we want our agent to learn is:

- to choose the minimum number of reachable stations necessary to reach the goal
- to minimize the route duration and associated cost.

To ensure that the agent effectively minimizes the number of stations chosen, the reward function doesn't simply provide positive rewards for valid actions. Such rewards could encourage the agent to take more steps than necessary to maximize rewards from reachable stations. Instead, the reward function assigns less negative rewards for valid actions, meaning that while they still result in a negative reward, the penalty is not as severe as the negative rewards for invalid actions. This approach encourages the agent to select reachable stations for a less negative reward and promotes choosing the minimum number of stations to reach the goal, as adding unnecessary stops leads to more negative rewards.

The reward for a valid action is composed of two components: an action component that assesses the action's quality on its own, and an information component that considers the action in the context of the previous path. The action component combines the time and cost components of the route between the current agent state and the state after the chosen action. This combination is defined by Equation 6.

$$Action\_component = c_{charging} + (t_{driving} + t_{charging})(1-\gamma) \tag{6}$$

The other component is a path component that consists of the sum of past action rewards up to the current agent state.

$$Path\_component = \sum reward \tag{7}$$

considering that our algorithm's objective is to minimize the route duration and cost, by utilizing the combination of these components our algorithm would be looking for the most time and cost consuming route. Luckily for this algorithm, due to the fact that we need to use negative rewards, by rewarding the agent with the negative combination of these components, the algorithm will as intended minimize the route's duration and cost.

However, with this reward function, we still have a problem, the agent has no incentive to look for the goal. Therefore in order to incentivize the agent to look for the goal, when the agent performs a valid action that archives the goal a supplementary positive reward is given to the agent.

$$Reward\ Function(action) = \begin{cases} reward = -(Path\_componment + Action\_component) & ,if \ \begin{matrix} action = valid \\ goal = not\ reached \end{matrix} \\ reward = (-(Path\_componment + Action\_component)) + goal\ reward & ,if \ \begin{matrix} action = valid \\ goal = reached \end{matrix} \\ reward = -Invalid\_action\_penalty & ,if\ action = not\ valid \end{cases} \tag{8}$$

With all this, our final reward function consists of the expression presented in Equation 8.

3) Deep Q Network architecture: One of the most important parts of the DQL algorithm is its q value aproximator. For the estimation of our Q-value for our DQL, we needed to design our Deep Q-Network (DQN), the neural network that will be our Q-value function approximator.

Neural Network

For the design of this DQN, we used a neural network consisting of 3 fully connected layers. The first layer has 1024 neurons with a ReLU activation function, the second layer has 256 neurons with a ReLU activation function, and in the final layer, the output layer, the number of neurons is equal to the number of possible actions of the agent and a linear activation function. The size of this neural network input is equal to the size of the environment observation, as for each step the agent feeds its environment observation to the neural network.

Dueling Networks

Due to the complexity of our environment, we used a dueling network architecture to improve the convergence speed of our DQL implementation. The decomposition of the Q-value into two networks is useful to better evaluate the quality of different actions that lead to similar rewards [11]. This improves significantly the convergence speed of our algorithm, especially in a complex environment with a high-action space like ours.

Epsilon greedy policy

In order for our agent to explore its environment, we need to give them an exploration strategy. For our implementation, we utilized the Epsilon Gready policy with a decaying Epsilon value, in order to explore our environment more efficiently.

III. Experimental Results

In this chapter, we present and analyze the results obtained in the experiments performed during this work.

A. Experimental Setup

In order to validate the behavior of our algorithms, an experimental setup needs to be designed.

Cost function Gamma

In order to solve the problem formulated in section II-A, we need to select a gamma value to combine the two costs. From our testing we decided to use a gamma value of 0.5 as this allows us to achieve a good balance between time optimization and cost optimization

1) Routes: In order to compare the different algorithms our experimental setup consisted of three routes. A short route from the city of Lisboa to the city of Covilhã, this trip's fastest road route has a distance of 277km, contains 265 possible charging stations and requires at least 1 recharging operation to be completed. A medium route from the city of Beja to the city of Porto, this trip's fastest road route has a distance of 448km, contains 463 possible charging stations, and requires at least 2 charging operations to be completed. And finally, a long route from the city of Bragança to the city of Vila Real de Santo António (VRSA), this trip's fastest road route has a distance of 778km, this route has 548 possible charging stations and requires at least 5 recharging operations to be completed.

TABLE II
GA experimental results parameters.

Parameters	values
max number of generations	150
Population size	15
Generations with constant fitness limit	50
Gamma(γ)	0,5
Crossover rate	0,7
Mutation rate	0,2

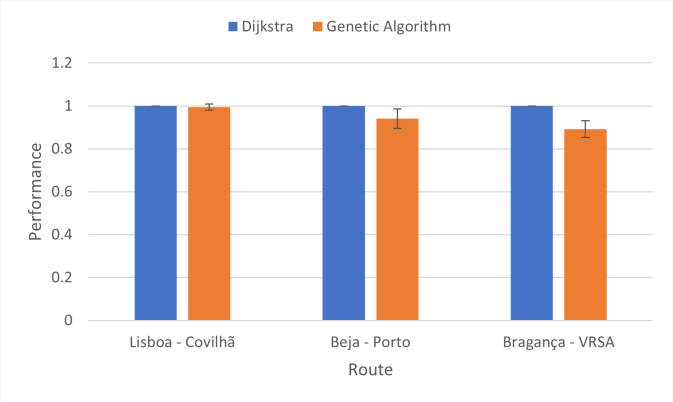

Fig. 3. Comparison of the GA performance results against the baseline results.

B. Results Obtained

1) Results Genetic Algorithm: To compare the performance of the GA to our baseline, we tested its performance using the parameters in Table II.

Table IV presents the results obtained performing the three routes described in section III-A1.

From the results in Table IV and figures 3 and 4, we can see that the GA can reliably achieve results very close to the optimal solution in a very reasonable runtime.

In Figure 3, we can observe the performance of the GA against the performance obtained from the Dijkstra's algorithm.

From the observation of Figure 3 and 4, we can conclude that the GA is able to achieve a very reliable and close to optimal solution in a fraction of the runtime of the Dijkstra's algorithm. These results show the strength and the potential the GA has to solve this problem, especially in very time-sensible applications.

2) Results Deep Q-Learning: We tested the behavior of the DQL algorithm in the three selected routes to compare its performance to the Dijkstra's algorithm and the Genetic algorithm's performance.

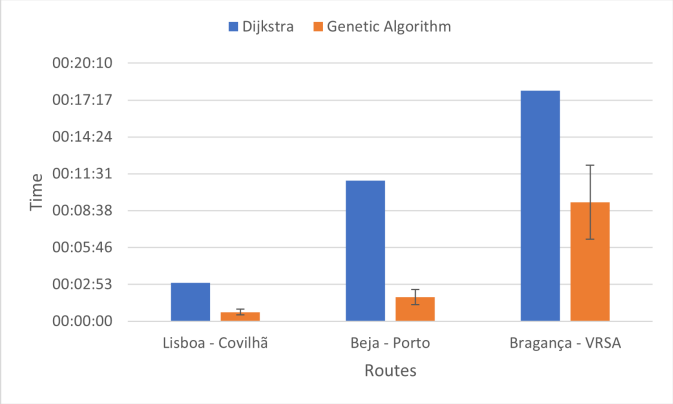

Fig. 4. Comparison of the GA runtime results against the baseline results.

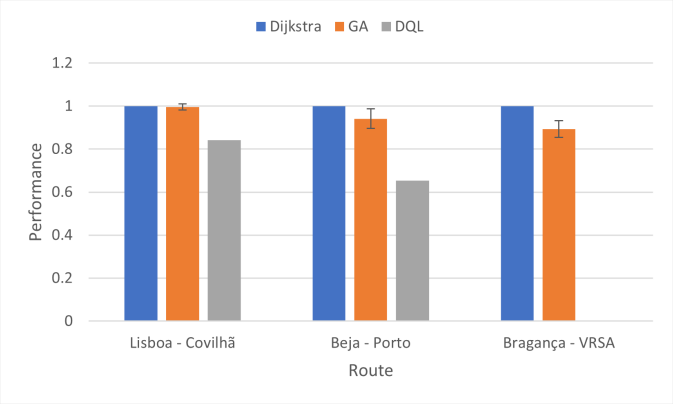

Fig. 5. Comparison of the performance results against the baseline results.

TABLE III
DQL EXPERIMENTAL RESULTS PARAMETERS.

Parameters	**values**
max number of steps per episode	10
starting epsilon(ϵ)	0,9
Min epsilon(ϵ) value	0,01
Epsilon(ϵ) decay rate	5e-5
Sequential memory size	5000
DQN optimizer	Adam
DQN learning rate	1e-3
Training size	400 000

During training, we observed that for the routes with a higher number of possible charging stations, the agent's rewards took longer to stabilize. Due to the action space of the agent being directly correlated with the number of charging stations in the environment, this data demonstrates the difficulty the algorithm has in learning with the increase of its action space.

This increase in difficult learning can be observed in Table IV and in Figure 5, where, while the solution of the algorithm in the short trip achieves a reasonable performance, for the medium trip the performance of the algorithm is very poor, and for the long trip, the algorithm was unable to converge into a valid route during the 10-hour training.

In Figure 5 we can better see how the performance of the DQL algorithm compares to the other algorithms.

By analyzing the figure we can see that for the short trip the performance of the algorithm is about 15% worse than the other algorithms. For the medium trip, we can see that the algorithm can only achieve a solution that performs 35% worse than the optimal solution obtained with Dijkstra's algorithm. This drop in performance combined with the algorithm not being able to provide a valid solution for the long trip, shows that in order to obtain a quality solution the algorithm would need to be trained for a much longer period of time. However, the capability to provide solutions of reasonable quality for the smaller maps shows the potential of the algorithm. With these results, we can conclude that with proper training, the algorithm can achieve a reliable solution. This combined with the advantage of obtaining a solution instantly and being able to respond to real-time changes in the environment, gives the DQL algorithm very high potential to be used in more dynamic and time-sensible applications.

IV. CONCLUSION

In this work, we have analyzed different approaches to find the optimal route of an EV in order to minimize the route duration and also the recharging cost.

We started by exploring the most classical approach, of using the Dijkstra's algorithm a search-based algorithm widely used to solve routing problems. From our implementation, we observed that while the algorithm is more than capable of achieving good results in solving the problem, the time complexity of the algorithm makes it unsuited to solve medium to large routes in a reasonable runtime.

To improve on the results obtained from the search-based approach we explore the use of two other methodologies.

We first explored the usage of Genetic Algorithms, a type of meta-heuristic approach. Meta-heuristic algorithms are utilized for their ability to find nearly optimal solutions within significantly reduced computation time compared to optimal algorithms like Dijkstra's algorithm. The results obtained from our implementation of the Genetic algorithm showed a great improvement in comparison to the Dijkstra's algorithm results. Obtaining near to optimal solutions in a fraction of the runtime. These findings show the potential of these algorithms for time-sensitive applications.

TABLE IV
RESULTS OBTAINED FROM THE THREE TESTED ALGORITHMS

	Short Trip				Medium Trip				Long Trip			
	Fitness	Time	Cost	Runtime	Fitness	Time	Cost	Runtime	Fitness	Time	Cost	Runtime
Dijkstra	0,4119	2:59:23	1,86	0:03:28	0,181	5:43:15	5,28	0:11:00	0,089	10:06:27	12,15	0:18:01
Genetic Agorithm	0,4100	3:01:01	1,86	0:00:43	0,171	5:55:13	5,79	0:01:53	0,080	10:44:23	14,23	0:09:18
(std dev)	(0,006)	(0:04:39)	(0,0007)	(0:00:14)	(0,008)	(0:20:08)	(0,423)	(0:00:35)	(0,003)	(0:31:08)	(0,89)	(0:02:54
Deep Q-learning	0,3466	3:21:00	2,42	-	0,118	6:56:24	9,9	-	DNC	DNC	DNC	-

Finally, we explored the application of the Deep Q-Learning algorithm, a prominent reinforcement learning technique. Deep Q-Learning is particularly suited for complex environments, as it leverages deep learning to model intricate, nonlinear relationships. However, the results we found in our implementation, showed a very poor performance in comparison to the results obtained from the other algorithms. These results where mainly due to the very high action space of our environment which required much more extensive training in order for the DQN to be able to better approximate the Q-values for our environment. Nevertheless, our initial testing of the algorithm showed that after adequate training the algorithm is able to achieve reliable solutions in real time. This capability to obtain a solution instantly is incredibly powerful and shows the potential this algorithm can achieve with the appropriate training.

In conclusion, our study demonstrates the trade-offs and advantages of different route optimization approaches. The choice of method depends on the specific requirements of the task, with traditional algorithms like Dijkstra's offering simplicity but limited scalability, while meta-heuristic approaches like Genetic Algorithms provide efficiency gains. Deep Q-Learning, although promising, requires substantial training but offers the potential for real-time solutions in complex scenarios.

REFERENCES

[1] *Global EV Outlook 2020: entering the decade of electric drive?* Paris: IEA Publications, 2020, oCLC: 1232229382.

[2] J. Sanguesa, V. Torres, P. Garrido, F. Martinez, and J. Marquez-Barja, "A review on electric vehicles: Technologies and challenges," *Smart Cities*, vol. 4, pp. 372–404, 03 2021.

[3] "Is This The Future? Waiting To Charge At Tesla Supercharger Station." [Online]. Available: https://insideevs.com/news/580338/is-this-future-waiting-charge-tesla-supercharger-station/

[4] K. Connolly, "Soaring energy costs could threaten future of electric cars, experts warn," *The Guardian*, Sep. 2022. [Online]. Available: https://www.theguardian.com/environment/2022/sep/12/soaring-energy-costs-could-threaten-future-of-electric-cars-experts-warn

[5] S. Schoenberg and F. Dressler, "Reducing Waiting Times at Charging Stations with Adaptive Electric Vehicle Route Planning," *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9672714/

[6] C. Liu, M. Zhou, J. Wu, C. Long, and Y. Wang, "Electric Vehicles En-Route Charging Navigation Systems: Joint Charging and Routing Optimization," *IEEE Transactions on Control Systems Technology*, vol. 27, no. 2, pp. 906–914, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8126871/

[7] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, Dec. 1959. [Online]. Available: http://link.springer.com/10.1007/BF01386390

[8] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3-4, pp. 279–292, May 1992. [Online]. Available: http://link.springer.com/10.1007/BF00992698

[9] K.-B. Lee, M. A. Ahmed, D.-K. Kang, and Y.-C. Kim, "Deep Reinforcement Learning Based Optimal Route and Charging Station Selection," *Energies*, vol. 13, no. 23, p. 6255, Nov. 2020. [Online]. Available: https://www.mdpi.com/1996-1073/13/23/6255

[10] T. Qian, C. Shao, X. Wang, and M. Shahidehpour, "Deep Reinforcement Learning for EV Charging Navigation by Coordinating Smart Grid and Intelligent Transportation System," *IEEE Transactions on Smart Grid*, vol. 11, no. 2, pp. 1714–1723, Mar. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8845652/

[11] Y. Zhang, M. Li, Y. Chen, Y.-Y. Chiang, and Y. Hua, "A Constraint-based Routing and Charging Methodology for Battery Electric Vehicles with Deep Reinforcement Learning," *IEEE Transactions on Smart Grid*, pp. 1–1, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9924526/

[12] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling Network Architectures for Deep Reinforcement Learning," 2015. [Online]. Available: https://arxiv.org/abs/1511.06581

[13] Y. Zhang, Q. Ma, X. Zhang, M. Gao, P. Yang, H. He, X. Hu, and B. Aliya, "Integrated Route and Charging Planning for Electric Vehicles Considering Nonlinear Charging Functions," in *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. Nanjing: IEEE, May 2018, pp. 660–665. [Online]. Available: https://ieeexplore.ieee.org/document/8465298/

[14] S. Shao, W. Guan, B. Ran, Z. He, and J. Bi, "Electric Vehicle Routing Problem with Charging Time and Variable Travel Time," *Mathematical Problems in Engineering*, vol. 2017, pp. 1–13, 2017. [Online]. Available: https://www.hindawi.com/journals/mpe/2017/5098183/

[15] C. Li, Y. Zhu, and K. Y. Lee, "Route Optimization of Electric Vehicles Based on Reinsertion Genetic Algorithm," *IEEE Transactions on Transportation Electrification*, vol. 9, no. 3, pp. 3753–3768, Sep. 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10025751/

[16] J. Barco, A. Guerra, L. Muñoz, and N. Quijano, "Optimal Routing and Scheduling of Charge for Electric Vehicles: A Case Study," *Mathematical Problems in Engineering*, vol. 2017, pp. 1–16, 2017. [Online]. Available: https://www.hindawi.com/journals/mpe/2017/8509783/

[17] D. Kosmanos, L. A. Maglaras, M. Mavrovouniotis, S. Moschoyiannis, A. Argyriou, A. Maglaras, and H. Janicke, "Route Optimization of Electric Vehicles Based on Dynamic Wireless Charging," *IEEE Access*, vol. 6, pp. 42 551–42 565, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8402042/

[18] "Openrouteservice." [Online]. Available: https://openrouteservice.org/

[19] "OpenStreetMap." [Online]. Available: https://www.openstreetmap.org/

[20] "Open Charge Map - Electric Vehicle Charging Locations Near You." [Online]. Available: https://map.openchargemap.io/