Multi-Factor Edge-Weighting with Reinforcement Learning for Load Balancing of Electric Vehicle Charging Stations

Lucas Hartman*, Santiago Gomez-Rosero†, Patrick Adjei‡, Miriam A. M. Capretz§
Department of Electrical and Computer Engineering
Western University, London, Ontario, Canada, N6A 5B9
Email: *lhartma8@uwo.ca, †sgomezro@uwo.ca, ‡padjei@uwo.ca, §mcapretz@uwo.ca

***Abstract*—The number of electric vehicle (EV) owners continues to grow at a rate that makes it increasingly difficult to avoid overloading the capacity of charging stations. Existing solutions to balance this load primarily focus on solving parts of the problem and fail to consider a more holistic approach that can solve the issue of balancing the load across multiple stations by handling EV routing. This paper proposes a novel solution that routes EVs while considering both travel times and peak loads at charging stations. The approach integrates a reinforcement learning algorithm with a path-finding algorithm and a custom-built simulation environment. Compared with baseline methods, the solution showed improved performance in minimizing peak power loads across charging stations while increasing individual trip duration by less than 5%. This approach has the potential to significantly improve the efficiency of EV charging by reducing peak loads at charging stations.**

***Index Terms*—Electric Vehicle, Reinforcement Learning, Path-finding Algorithms, Load Balancing**

I. Introduction

The adoption of electric vehicles (EVs) continues to increase in Canada, and with Canada set to ban the sale of internal combustion engines by 2035 [1], there seems to be no intent to stop this new trend. Although EVs have a positive environmental impact [2], they lack the existing and mature gas station grid that supports their alternative. This creates the challenge of finding the best way to balance the increasingly high demand for power across the still-developing charging station infrastructure.

To address this challenge, this work proposes a novel approach to simplify the EV charging process and driving experience while avoiding clustering large numbers of EVs at individual stations. The proposed solution is a technique called multi-factor edge-weighting with reinforcement learning (MERL), which leverages reinforcement learning (RL) strategies to enhance an existing path-finding algorithm. It does this by adjusting the parameters involved in building the weighted graph that is used to model the simulation environment. MERL provides routing recommendations that optimize various factors such as travel time, distance, peak power loads at charging stations, and waiting times at charging stations.

The application of MERL integrates two key considerations: the vehicle routing problem (VRP), which involves optimizing routes between multiple destinations with the goal of minimizing total route cost [3], and the charge scheduling problem (CSP), which involves determining the best times to charge EVs so that the cost and the strain on the electrical grid are minimized [4]. By tackling these issues concurrently, the proposed approach aims to significantly enhance the overall EV user experience.

It is important to address the integration of charge scheduling into the EV routing problem, not only because this will bolster the ability of the existing infrastructure to handle increasing EV adoption, but also because it could be used to enforce protocols that route traffic among stations to benefit involved parties beyond the drivers themselves. Currently, the standard process for selecting a charging station involves the user selecting from a set of recommended stations based on how far away these stations are, how many chargers are available for use, and the output rate of the available chargers. The decision on which option is the most time-efficient is left to the driver, and the focus of this selection process is individual convenience. This makes it hard to predict traffic at each station because drivers must handle load balancing themselves. Should the proposed solution be adopted, it would cause more predictable and uniform behavior among EV drivers, shorter wait times at stations, and increase the stability of the charging stations supporting the EVs.

The contribution of this paper is a novel method to route EVs that balances traffic among several charging stations while increasing the average distance of each trip by no more than 5% compared to the baseline alternatives in the simulation environment used. This solution could be deployed as an individual EV routing algorithm or could be used to direct swarms of EVs in a way that avoids clustering at charging stations without inconveniencing drivers.

This paper is organized as follows: Section II describes the components used by the proposed method. Section III examines existing approaches to addressing the VRP and CSP. Section IV describes the methodology for implementing MERL, and section V presents the results obtained. Section VI discusses the findings and their implications and section VII concludes the paper.

II. Background

This section introduces and explains the main concepts involved in the application of MERL.

A. Reinforcement Learning

Reinforcement learning (RL) is a sub-field of machine learning where agents learn to make decisions by interacting with an environment [5]. Unlike supervised learning, where the agent is explicitly taught, in RL, the agent learns from the consequences of its actions. These consequences are quantified by means of a reward function which is used to evaluate the quality of an action in a state.

Formally, an RL problem is modeled as a Markov decision process (MDP). An MDP is a tuple (S, A, P, R, γ), where:

- S is the state space, representing the different configurations of the environment.
- A is the action space, enumerating the possible actions that the agent can take.
- $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability function, which defines the dynamics of the environment. Specifically, $P(s'|s, a)$ is the probability of transitioning to state s' given that the agent took action a in state s.
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function, which provides a scalar feedback signal to the agent. $R(s, a, s')$ is the expected reward received after transitioning to state s' from state s due to action a.
- $\gamma \in [0, 1]$ is the discount factor which determines the present value of future rewards.

In traditional RL usage, the agent's behavior is defined by a policy $\pi : S \times A \rightarrow [0, 1]$, which specifies the probability $\pi(a|s)$ that an agent will take action a when in state s. The goal of the agent is to learn an optimal policy π^* that maximizes the expected cumulative discounted reward [6].

In this paper, Deep Reinforcement Learning (DRL), a sub-field of RL that uses deep learning techniques to handle complex environments with large state-action spaces, is leveraged. DRL is particularly useful when the state and/or action spaces are high-dimensional, or when the state is represented by raw sensory data, such as images.

A key concept in RL and DRL is the state-action value function, commonly referred to as the Q-function and denoted as $Q^\pi(s, a)$. This function estimates the expected return if the agent starts in state s, takes action a, and follows policy π thereafter. The Q-function is instrumental in deriving the agent's policy, and the accuracy of the Q-function is indicative of the agent's performance. The optimal Q-function $Q^*(s, a)$ corresponds to the maximum expected return achievable by following any policy, given that the agent starts in state s and takes action a.

B. Deep-Q Learning

Deep-Q learning (DQL) is a variant of Q-learning that uses a deep neural network to approximate the Q-function which increases its ability to scale and generalize [7]. In DQL, the agent uses a neural network, known as the Q-network, to estimate the Q-values for each action given a state. The Q-network is trained to minimize the difference between the predicted Q-values and the target Q-values, which are computed using the Bellman equation in Eq. 1:

$$Q(s, a) = \mathbb{E}_{s' \sim \mathcal{E}}[r + \gamma \max_{a'} Q^*(s', a')|s, a], \tag{1}$$

where s' is the next state, r is the reward, γ is the discount factor, and $\mathcal{E}$ is the environment. The expectation $\mathbb{E}$ represents an average over the possible next states s', which are sampled from the environment $\mathcal{E}$. The notation $\mathbb{E}_{s' \sim \mathcal{E}}[...]|s, a]$ means that the expectation is taken with respect to the distribution of s' given the current state s and action a. This term essentially captures the agent's uncertainty about the next state due to the inherent stochasticity of the environment.

The Q-network is trained by minimizing the loss function shown in Eq. 2:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2], \tag{2}$$

where θ are the parameters of the Q-network, θ^- are the parameters of a target network that is periodically updated with the parameters of the Q-network, and $\mathcal{D}$ is a replay buffer that stores the agent's experiences (s, a, r, s').

The target network and the replay buffer are two key techniques in DQL. The target network helps stabilize the learning process by providing a fixed target for the Q-network. The replay buffer enables the agent to learn from past experiences, which improves sample efficiency and breaks the correlation between consecutive experiences.

DQL has been successfully applied to a variety of tasks, including playing Atari games [8], controlling robotic arms [9], and playing the game of Go [10]. This paper proposes to use DQL to enhance the performance of path-finding algorithms.

III. Related Work

Existing research [11][12][13] that addresses the VRP with EVs is limited in scope and fails to consider important factors such as the current load placed on a charging station's power supply and the traffic at charging stations. Previous attempts to solve this problem have primarily focused on minimizing the time an EV owner spends driving [14][15][16] or minimizing the distance that the driver travels [17][18][19]. Similarly, existing research addressing the CSP with EVs [20][21][22] fails to consider the opportunity to route an EV to a different station entirely to balance power loads across charging stations.

Tesla's "Trip Planner" is the closest comparison to the proposed approach and shares the same final implementation vision. However, it differs in that it is designed only to minimize the amount of time spent driving and charging [14]. When specifying a route, the Trip Planner will automatically include recommended stops along the path to ensure the driver can arrive at the destination without running out of energy. This solution does not consider factors related to each charging station, such as the traffic and the energy load it incurs. In

contrast, the application of MERL makes recommendations that include those additional factors to improve the experience for both the driver and the charging station operators.

As mentioned by Abid *et al*. [11], the challenges of routing and charging EVs are deeply connected, but existing approaches treat them separately to manage scope. The routing of vehicles is abstracted to the traveling salesmen problem [23] and the scheduling of EV charging is treated as a mixed-integer linear programming problem [24]. Machine learning techniques have been used to solve the VRP and CSP separately [20][25][26], but have not been applied to the combinations of the two.

Park and Moon [27] used RL to deal with the scheduling and allocation of EV charging, but did not consider the routing of the EVs to the allotted charging stations. Cai *et al*. [28] used Dijkstra's algorithm with a few performance enhancements to match drivers to charging stations that are close to them. Much like Tesla's Trip Planner, the main downside of this approach is that it also does not consider the load placed on the charging stations or the wait times caused by increased traffic at centralized stations. Lee *et al*. [29] used RL to select a charging station where an EV should stop when it falls below a battery threshold, but this approach requires the driver to adapt to a route change during the trip, instead of having the route change determined before the trip begins.

IV. Methodology

The proposed approach to addressing the fusion of the VRP and CSP involves a novel integration of RL with the building of weighted graphs that considers how multiple important factors influence the weights of a graph. Through training, the RL agent will work with a common pathfinding algorithm called Dijkstra's algorithm [30] to find the shortest path considering both the traffic at charging stations as well as the distance and time to traverse the path.

The first section details how RL is used to enhance Dijkstra's algorithm, the second section discusses how the integration is used in this paper to create the proposed solution, and the third section discusses the details of the reward function used during training.

A. Multi-factor Edge-weighting with Reinforcement Learning (MERL)

The methodology proposed in this paper uses a new approach for creating weighted graphs, called multi-factor edge-weighting with reinforcement learning (MERL). MERL converts an unweighted graph into a weighted one by taking a weighted sum of multiple factors within the environment, where the weights used in the summation are assigned by an RL agent. These weighted sums are then used to build the edge weights in the graph. Factors, in this context, are attributes that influence the value of the overall objective function. In this work, the factors that will be considered are the amount of traffic at each charging station, measured by the number of cars, and the physical distances to charging stations, measured in kilometers.

The value assigned to each individual factor will be referred to as the factor's impact rating (IR). To find the optimal IR for each factor, an RL algorithm will update them, calculate the edge weights based on the assigned ratings, and give the new weighted graph to a path-finding algorithm.

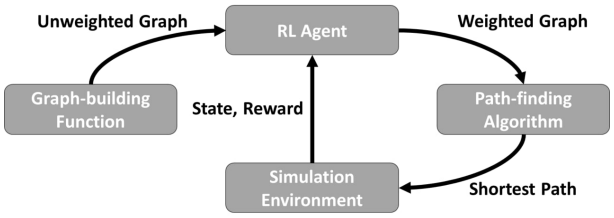

Fig. 1. Feedback loop for MERL.

Figure 1 illustrates the feedback loop that is used to train MERL to build an optimal weighted graph. Note that the graph-building function will create an unweighted graph once, and the RL agent will then effectively decide the weights of the graph by first deciding on the IRs and then using those IRs to calculate the weights. The weighted graph created by the RL agent is then given to a path-finding algorithm to solve. When the shortest path is computed by the path-finding algorithm, the simulation environment takes that path and simulates its traversal, outputting the next state and the reward obtained using the given path from which the RL agent learned.

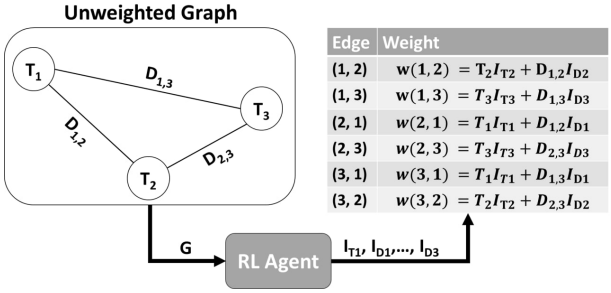

Fig. 2. Using MERL with distances and traffic levels.

Figure 2 shows how MERL converts an unweighted graph into a weighted one in the context of the proposed method. Each node i has a traffic level T_i, and each set of connected nodes (j, i) has a distance $D_{j,i}$. For each node, the RL agent assigns an IR to traffic level I_{Ti} and distance I_{Di} at node i. It then uses the IRs to obtain a weighted sum of the traffic level and distance, which becomes the edge weight $w(j, i)$ connecting nodes j and i, where $i \neq j$. For example, to find the cost of going from node 1 to node 2, it multiplies the IR for traffic (I_{T2}) by the traffic level at node 2 (T_2) and adds it to the product of the IR for distance (I_{D2}) with the between-node distance ($D_{1,2}$). The general formula for calculating an edge weight using MERL is shown in Eq. 3:

$$w(j, i) = \sum_{k=0}^{K} A_{k,i} I_{k,i}, \tag{3}$$

where K is the number of attributes considered, $A_{k,i}$ is the value of attribute A_k at node i, $I_{k,i}$ is the assigned IR of A_k at node i, and $w(j,i)$ is the calculated edge weight between nodes j and i.

Although the distance between nodes is symmetrically equivalent, the traffic levels are not. Because of this, MERL creates an asymmetric edge weight between the nodes in the graph. The IRs will also affect the edge weights in an asymmetric way.

Initially, the assigned IRs will be random for each attribute. The RL agent learns to refine them by repeatedly adjusting them, running a path-finding algorithm on the resulting graph, and being given the reward associated with the shortest path found using the assigned IRs.

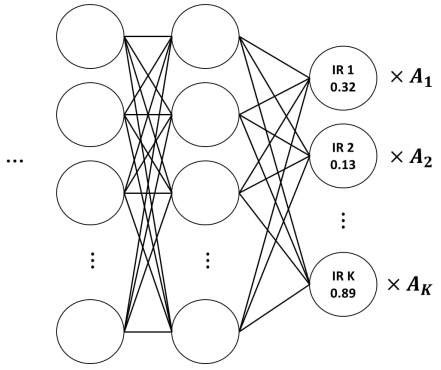

Fig. 3. Output structure of the neural network used in MERL.

Unlike traditional RL techniques that output an action for an agent to take, the output of the RL technique used in MERL is the IRs for each of the attributes. As shown in Fig. 3, each output neuron is an IR that takes a value between 0 and 1, which is multiplied by its associated attribute value. The formula to determine the number of outputs from the RL agent is shown in Eq. 4:

$$O_{size}(A,V) = KV, \tag{4}$$

where O_{size} is the number of outputs, K is the number of attributes, and V is the number of nodes in the graph. This equation shows that the RL agent needs an output value for all attributes at every node within the graph, which is used as the IR for that attribute at that node.

B. Application of Multi-factor Edge-weighting with Reinforcement Learning

The MERL approach has been applied to the problem of routing EVs while enforcing the constraint that each EV must stop at a charging station before going to its destination. The objective function considers multiple factors and aims to minimize both the time spent traveling and the peaks in traffic levels at charging stations.

As shown in Fig. 4, the graph used in this application of MERL has nodes representing the charging stations, as well as two nodes representing the starting and ending points of the route. Each edge of the graph indicates that it is possible to travel between both nodes on a fully charged battery, with the exception of the edges connected to the starting point which indicates that it is possible to travel along that edge using the starting battery level. This is because the edges originating from the charging stations allow the EV to charge the battery as high as needed at the charging station node, but the EV must be able to reach those charging stations on its starting battery from the origin node. The graph also has an optimal route which is selected by Dijkstra's algorithm given the weights assigned using MERL.

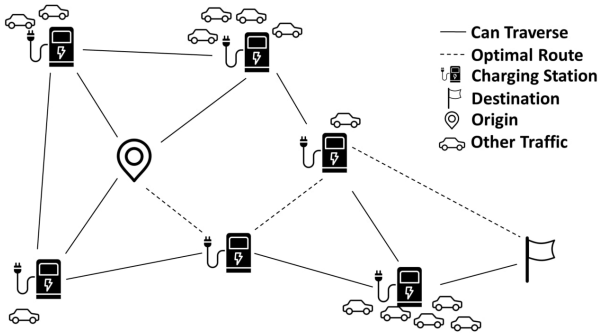

Fig. 4. Simulation environment.

Each station has a traffic and distance attribute, meaning that the RL agent must decide for each station how important the traffic level is compared to the distance. To decrease the duration of training the agent, the RL model outputs one value for every node instead of one value for every attribute. This works because only two attributes are considered in this application of MERL, meaning that the IR for traffic (I_{Tj}) assigned to node j can also be used to calculate the IR for the distance (I_{Dj}), such that $I_{Dj} = (1 - I_{Tj})$.

To ensure that each value in the output layer of the neural network is between 0 and 1, the output layer uses a sigmoid function to determine the final output values. Using the MERL approach, the IRs assigned by the agent are then used to compute the edge weights of the graph that represents the possible routes an EV can take to go from its origin to its destination. Dijkstra's algorithm will take the weighted graph and find the shortest path based on the edge weights determined by the RL agent. This shortest path is then simulated in the environment, and the reward associated with that path is returned to the agent.

DQL is the RL technique used to learn the optimal IRs. The state information given to the agent contains the total number of chargers, the total distance from origin to destination, the number of EVs within the simulation, the traffic level at each charging station, and the distances of each charging station from the origin.

The simulation environment allows an EV to select from the list of all nodes to travel towards and moves the EV along the given path in fixed intervals. For each timestep in the simulation, the EV can either move toward one of the nodes in a straight line or charge at one of the charging stations.

When an EV is not charging at a station, its battery level will decrease proportionately to how far it has traveled.

The path given to the environment after using Dijkstra's algorithm is a set of steps consisting of the destination to travel to in that step and the battery level needed before going to the next step. The simulation environment goes through each step by traveling towards the indicated destination and charging to the indicated amount until it reaches the final step, which is the route destination.

When charging at a station, the station will increase the battery level of the EV by its maximum discharge rate. As in other research [31], the discharge rate is fixed until the cumulative output across all chargers exceeds the maximum output threshold of the entire station. This is shown in Eq. 5:

$$C_{out}(T,t,c) = \min\left(\frac{C_{ind}}{T_{t,c}}, C_{ind}\right), \tag{5}$$

where C_{out} is the charge output of the station to an individual EV, C_{ind} is the discharge rate of an individual charger at the station, and $T_{t,c}$ is the number of EVs charging at station c at timestep t.

To simulate the traffic at charging stations, this approach is applied to an environment with multiple EVs. Each EV has a route that it needs to traverse, and each EV cannot make it to its destination without stopping to charge at least once. When an EV chooses to charge at a station, it increases that station's traffic level by one when it arrives and decreases it by one when it leaves. The path-finding algorithm will find the shortest paths for the multiple EVs in the environment by determining their paths in sequential order. This means that when finding the path for the first EV, the traffic at each station is 0, which enables it to simply take the route with the shortest distance. Because each EV must stop at least once on its path, the second EV will make a decision with the knowledge that one station has a traffic level of 1. This continues until the final EV decides on a route with the knowledge of where every other EV will stop in the simulation. In a deployed version of this approach, the traffic considered could either be the estimated traffic at the time of arrival or the current traffic at the time the route is determined.

The number of chargers available for each EV to stop at is fixed and consistent across all routes. The distance of each route and the starting battery level of each EV varies by EV to give a more accurate representation of the real-world environment. However, the starting battery level must be low enough to prevent the EV from traveling directly to its destination, because otherwise it would have no need to stop at a station.

C. The Reward Function

To train the RL agent in the process of applying MERL, an objective function is used, which represents the overall goal that the agent is trying to achieve. In this case, that goal is to minimize the time spent traveling, the peaks in traffic, and the distance traveled. The reward is calculated for each timestep in the simulation, and the sum of these individual rewards is returned to the RL agent after the end of the simulation. The reward function used to train the RL agent is shown in Eq. 6:

$$R = -\sum_{t=0}\left(\frac{D_{t,rem}}{D_T} + \max_c T_{t,c} + \sum_{h=0}^{t} M_h\right), \tag{6}$$

where $D_{t,rem}$ is the current distance remaining to the destination at timestep t, D_T is the total distance from origin to destination, $T_{t,c}$ is the traffic at charging station c at timestep t, and M_h is the distance traveled at timestep h in kilometers.

Because the reward is negative and is calculated at each timestep, the agent maximizes the total reward by minimizing the number of steps taken to the destination. To maximize the reward for each timestep, the agent must minimize the distance between its current position and the destination while also minimizing the peak amount of traffic at any station and the cumulative distance traveled. This reward function is how the agent learns to minimize trip duration, peaks in traffic levels at stations, and distance traveled.

V. Evaluation

To evaluate the proposed method, an experimental setup was created that simulates using MERL to route a swarm of EVs in an urban setting. The details of the experimental setup and the results obtained through the experiment are described in this section.

A. Experimental Setup

The agent was given control of an environment where 100 EVs had to find the shortest path for their unique and distinct origin and destination points, which were between 10 and 20 km apart, or 90%-180% of London, Ontario's estimated radius [32]. To compare the proposed method to alternative baseline approaches, the simulation was run using four cases across three seeds.

The agent was trained for 100 episodes using epsilon-greedy exploration [33], where each episode involved the agent routing all 100 EVs, giving the agent 10,000 experiences to learn from. The training process took 2 hours for each seed using a desktop powered by an NVIDIA GeForce RTX 3080 and an AMD Ryzen 5 1600X Six-Core Processor. Each seed is trained individually before the agent is evaluated on it. Once trained, the model takes less than 5 seconds to process input and generate the weighted graph.

The agent and environment were programmed using Python with the PyTorch and numpy libraries. The values of the hyperparameters used are found in Table I.

TABLE I
HYPERPARAMETERS

Parameter	Description	Value
α	Learning Rate	0.0001
ϵ	Exploration Probability	0.8
γ	Discount Factor	0.9999
ϵ-decay	Rate of Decay for Epsilon	0.9

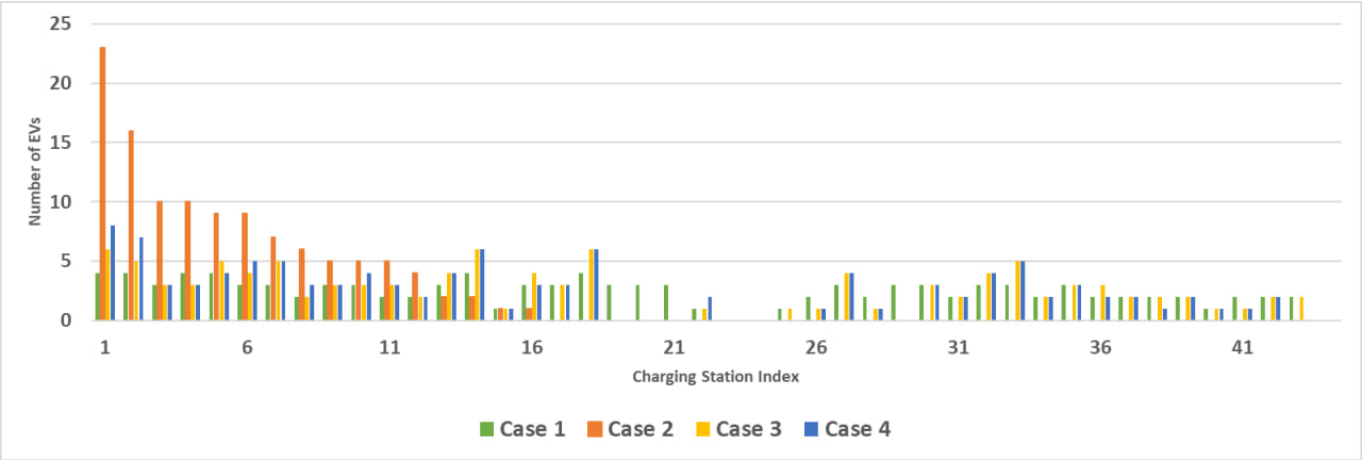

Fig. 5. Peak traffic level at each charging station.

B. Results

In case 1, Dijkstra's algorithm was applied considering only traffic levels; in case 2, Dijkstra's algorithm was applied considering only distances. Case 3 applied Dijkstra's algorithm giving equal consideration to both traffic levels and distances, and case 4 used MERL to find a balance between the impacts of traffic levels and distances. Cases 1-3 are the baselines for comparisons with the proposed solution, and case 4 is the proposed method itself.

Each EV had the same fixed number of chargers available to it, which were based on real-world data. The set of chargers contained the closest chargers to the EVs' origin, destination, and midpoint. To populate the charging stations in the environment, a dataset of every public charging station in Ontario was supplied by National Renewable Energy Laboratory [34], along with information pertaining to these charging stations, including their geographical coordinates. Each EV had its own starting battery percentage, which ranged from 13%-15% of the total battery level but was restricted from going below 10%. This ensured that each EV would be required to stop at a charging station before reaching its destination, but gave certain EVs the option of traveling to stations outside the set near the route origin.

Tables II, III, and IV show how the proposed method compared to the various baselines. Note that the values are taken across all three seeds. Table II compares the performance of the four cases at balancing traffic levels across stations, Table III compares the travel times of the four cases, which includes the time spent driving and charging, and Table IV compares the distances traveled in each case.

Table II shows that case 4 using MERL reduced peak traffic level by 65% compared to case 2, which aimed to minimize distance traveled. Figure 5 demonstrates that MERL evenly balanced traffic across a large number of stations, whereas case 2 clustered traffic across a small number of stations.

Table III shows that the number of timesteps taken varied little across all cases, but that case 2, which aimed only to minimize distance, took the longest time to complete due to the increase in time spent at charging stations. This shows how existing approaches to minimize the time on the road can actually increase the total time spent traveling.

Table IV shows that case 4 using MERL decreased the maximum distance by 45% compared to case 1 which focused only on balancing traffic. This shows why although case 1 can effectively balance traffic loads, it is not a viable solution because the routes chosen would be too inconvenient for the driver. Case 4, however, only increased the maximum trip distance by less than 2.5 km and the average trip distance by less than half a kilometer compared to case 2, which had

TABLE II
COMPARISON OF TRAFFIC DURING SIMULATION

	Avg Traffic	Max Traffic
Case 1 (Traffic Focused)	0.425 EVs	4.000 EVs
Case 2 (Distance Focused)	0.432 EVs	23.000 EVs
Case 3 (Evenly Balanced)	0.451 EVs	6.000 EVs
Case 4 (MERL Integrated)	0.445 EVs	8.000 EVs

TABLE III
COMPARISON OF TIME SPENT TRAVELING IN SIMULATION

	Avg Steps	Max Steps
Case 1 (Traffic Focused)	23.860 steps	38.000 steps
Case 2 (Distance Focused)	24.973 steps	42.000 steps
Case 3 (Evenly Balanced)	23.960 steps	35.000 steps
Case 4 (MERL Integrated)	23.667 steps	35.000 steps

TABLE IV
COMPARISON OF DISTANCE TRAVELED IN SIMULATION

	Avg Dist	Max Dist
Case 1 (Traffic Focused)	26.881 km	58.700 km
Case 2 (Distance Focused)	20.330 km	30.000 km
Case 3 (Evenly Balanced)	21.228 km	36.120 km
Case 4 (MERL Integrated)	20.774 km	32.470 km

the best distance metrics, but clustered traffic. MERL also performed better than case 3, which evenly balanced the focus between traffic and distance, by 11%.

Lastly, Fig. 6 shows the reward function throughout the training of the RL agent across the three seeds. The solid green line is the average across the three seeds, and the surrounding area contains the minimum to maximum values. In all three seeds, the training converged to a stable reward value around episode 50. Note that each episode contained 100 EVs and therefore 100 learning experiences, so the agents converged after roughly 5000 training experiences.

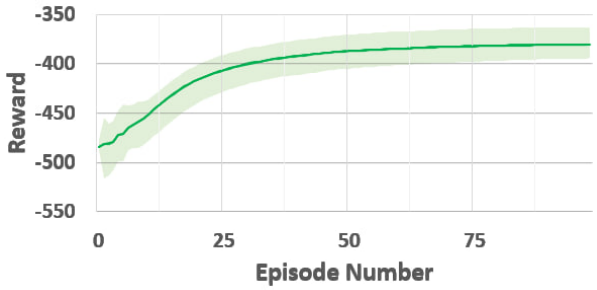

Fig. 6. Reward function during training.

VI. Discussion

The results of this study indicate that the proposed approach can effectively balance the load across charging stations while also providing a satisfactory user experience for EV owners. MERL eliminated peaks in traffic without increasing the average trip distance by more than 3% compared to a baseline emulating the current industry standard. This has important implications for the adoption of EVs and the stability of charging stations. By using an RL approach, the charging process for EVs can be optimized, and the output strain on charging stations can be reduced.

A perfect approach to address the vehicle routing and charge scheduling problems will be subjective for the person routing the trips. Some drivers may prioritize minimizing travel time, whereas others may prioritize minimizing charging costs, or something else entirely. The proposed methodology could address these preferences by updating the reward function in Eq. 6 to place more emphasis on either distance or traffic level by multiplying the ratios that represent their importance. The reward function could also be updated to consider more factors such as trip cost, resulting in a highly adaptable approach to addressing the VRP and CSP.

VII. Conclusion

This paper created a novel method called multi-factor edge-weighting with reinforcement learning (MERL) which is used to handle the routing and charging of EVs by combining RL and Dijkstra's algorithm. The MERL method was then evaluated using a simulation environment where it demonstrated its effectiveness at balancing the load placed on charging stations by large groups of EVs.

MERL integrates the charging process into the routing process, providing a simple and convenient experience for EV drivers without increasing average travel times by more than 5% compared to a baseline emulating the current industry standard. With a more convenient charging process for EV owners, there is also less reason for people to avoid switching from traditional combustion vehicles to EVs, which could have significant environmental benefits.

Although the results of this study are promising, several areas remain for future research. The simulation environment used in this study is relatively simple and does not fully capture the complexity of a real-world city. Future research could involve developing a more realistic simulation environment including factors such as road traffic congestion, variable charging station availability, and more complex routes to estimate the effectiveness of MERL in a real-world implementation, which could later be verified through a deployed implementation. Future research could also involve increasing the scope of the tests to more varied route distances and starting battery levels.

The MERL method was evaluated on only the vehicle routing and charge scheduling problems; however, because it is essentially just a way to find weights for a graph when balancing multiple factors, it could theoretically be used in other applications that require weighted graphs. Fields such as networking, robotics, social media, and video games all use path-finding algorithms and weighted graphs and could therefore potentially benefit from the integration of the MERL method. Future research could include applying the MERL method to these other applications to see how well it generalizes to other problems.

Overall, the integration of RL as a solution to the wide field of EV problems has ample potential. RL is already being used to address challenges such as autonomous driving [35], improving battery life [36], and many more. This paper has shown that integrating it into the routing process could also prove to be useful by taking over the decision-making process of finding where to charge.

References

1. Fingas, J. Canada will ban sales of combustion engine passenger cars by 2035. https://www.engadget.com/canada-combustion-engine-car-ban-2035-154623071.html (2022).
2. Sivagnanam, A. *et al.* Minimizing energy use of mixed-fleet public transit for fixed-route service. in *Proceedings of the AAAI Conference on Artificial Intelligence* **35** (2021), 14930–14938.
3. Laporte, G. What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)* **54,** 811–819 (2007).
4. Zhang, T., Chen, W., Han, Z. & Cao, Z. Charging scheduling of electric vehicles with local renewable energy under uncertain electric vehicle arrival and grid power price. *IEEE Transactions on Vehicular Technology* **63,** 2600–2612 (2013).

5. Wiering, M. A. & Van Otterlo, M. Reinforcement learning. *Adaptation, learning, and optimization* **12,** 729 (2012).
6. Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science* **2,** 331–434 (1990).
7. Mnih, V. *et al.* Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
8. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *nature* **518,** 529–533 (2015).
9. Zhang, S., Yao, L., Sun, A. & Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM computing surveys (CSUR)* **52,** 1–38 (2019).
10. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *nature* **529,** 484–489 (2016).
11. Abid, M., Tabaa, M., Chakir, A. & Hachimi, H. Routing and charging of electric vehicles: Literature review. *Energy Reports* **8,** 556–578 (2022).
12. Sweda, T. M., Dolinskaya, I. S. & Klabjan, D. Adaptive routing and recharging policies for electric vehicles. *Transportation Science* **51,** 1326–1348 (2017).
13. Desaulniers, G., Errico, F., Irnich, S. & Schneider, M. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research* **64,** 1388–1405 (2016).
14. Tesla. Tesla Trip Planner Documentation. https://www.tesla . com / ownersmanual / modelx / en_us / GUID - 01F1A582 - 99D1 - 4933 - B5FB - B2F0203FFE6F . html (2023).
15. Li, J., Wang, F. & He, Y. Electric vehicle routing problem with battery swapping considering energy consumption and carbon emissions. *Sustainability* **12,** 10537 (2020).
16. Shao, S., Guan, W., Ran, B., He, Z., Bi, J., *et al.* Electric vehicle routing problem with charging time and variable travel time. *Mathematical Problems in Engineering* **2017** (2017).
17. Li, G., Sun, Q., Boukhatem, L., Wu, J. & Yang, J. Intelligent vehicle-to-vehicle charging navigation for mobile electric vehicles via VANET-based communication. *IEEE Access* **7,** 170888–170906 (2019).
18. Zhao, Z., Li, X. & Zhou, X. Distribution route optimization for electric vehicles in urban cold chain logistics for fresh products under time-varying traffic conditions. *Mathematical Problems in Engineering* **2020,** 1–17 (2020).
19. Deb, S. *et al.* A robust two-stage planning model for the charging station placement problem considering road traffic uncertainty. *IEEE Transactions on Intelligent Transportation Systems* **23,** 6571–6585 (2021).
20. Dang, Q., Wu, D. & Boulet, B. A q-learning based charging scheduling scheme for electric vehicles. in *2019 IEEE Transportation Electrification Conference and Expo (ITEC)* (2019), 1–5.
21. Bayram, I. S., Michailidis, G. & Devetsikiotis, M. Unsplittable load balancing in a network of charging stations under QoS guarantees. *IEEE Transactions on Smart Grid* **6,** 1292–1302 (2014).
22. Shibl, M., Ismail, L. & Massoud, A. Electric vehicles charging management using machine learning considering fast charging and vehicle-to-grid operation. *Energies* **14,** 6199 (2021).
23. Little, J. D., Murty, K. G., Sweeney, D. W. & Karel, C. An algorithm for the traveling salesman problem. *Operations research* **11,** 972–989 (1963).
24. Vielma, J. P. Mixed integer linear programming formulation techniques. *Siam Review* **57,** 3–57 (2015).
25. Nazari, M., Oroojlooy, A., Snyder, L. & Takác, M. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* **31** (2018).
26. Li, H., Wan, Z. & He, H. Constrained EV charging scheduling based on safe deep reinforcement learning. *IEEE Transactions on Smart Grid* **11,** 2427–2439 (2019).
27. Park, K. & Moon, I. Multi-agent deep reinforcement learning approach for EV charging scheduling in a smart grid. *Applied Energy* **328,** 120111 (2022).
28. Cai, J., Chen, D., Jiang, S. & Pan, W. Dynamic-area-based shortest-path algorithm for intelligent charging guidance of electric vehicles. *Sustainability* **12,** 7343 (2020).
29. Lee, K.-B., A. Ahmed, M., Kang, D.-K. & Kim, Y.-C. Deep reinforcement learning based optimal route and charging station selection. *Energies* **13,** 6255 (2020).
30. Javaid, A. Understanding Dijkstra's algorithm. *Available at SSRN 2340905* (2013).
31. Bac, U. & Erdem, M. Optimization of electric vehicle recharge schedule and routing problem with time windows and partial recharge: A comparative study for an urban logistics fleet. *Sustainable Cities and Society* **70,** 102883 (2021).
32. Government of Canada, S. C. Census Profile, 2016 Census - London, City [Census subdivision], Ontario and Ontario [Province]. https://www12.statcan.gc.ca/census-recensement/2016/dp-pd/prof/details/page.cfm?Lang=E&Geo1=CSD&Code1=3539036&Geo2=PR&Code2=35&SearchText=london&SearchType=Begins&SearchPR=01&B1=All&TABID=1&type=0 (2017).
33. Paavai Anand, P. *et al.* A brief study of deep reinforcement learning with epsilon-greedy exploration. *International Journal Of Computing and Digital System* (2021).
34. Network, N. D. Alternative Fuel Stations API. https://developer.nrel.gov/docs/transportation/alt-fuel-stations-v1/all/ (2023).
35. Xu, P., Dherbomez, G., Héry, E., Abidli, A. & Bonnifait, P. System architecture of a driverless electric car in the grand cooperative driving challenge. *IEEE Intelligent Transportation Systems Magazine* **10,** 47–59 (2018).
36. Harper, G. *et al.* Recycling lithium-ion batteries from electric vehicles. *nature* **575,** 75–86 (2019).