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

Lucas Hartman*, Santiago Gomez-Rosero†, Patrick Adjei‡, Miriam Capretz§
Department of Electrical and Computer Engineering
Western University, London, Ontario, Canada, N6A 4H6
Email: *lhartma8@uwo.ca, †sgomezro@uwo.ca, ‡padjei@uwo.ca, §mcapretz@uwo.ca

***Abstract*—The amount of electric vehicle (EV) owners continue 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 the routing of EVs. This paper proposes a novel solution that routes EVs while considering both travel times and peak loads at charging stations. The approach used integrates a reinforcement learning algorithm with path-finding algorithms and a custom-built simulation environment. When compared with baseline methods the solution showed improved performance in minimizing peak power loads across the charging stations without causing an increase in individual trip duration over 5%. This approach has the potential to significantly improve the efficiency of EV charging by reducing the peak loads at charging stations.**

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

I. Introduction

The adoption of electric vehicles (EVs) continues to increase in Canada, and with Canada to ban the sale of internal combustion engines by 2035 [1] there seems to be no intent to stop this new standard. While EVs have a positive environmental impact [2], they lack the existing and mature gas station grid which supports their alternative. This creates a challenge of finding the best way to balance the increasingly large 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 quantities of EVs at individual stations. The proposed solution is a technique called multi-factor edge-weighting with reinforcement learning (MERL) that 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 the routes between multiple destinations with the goal of minimizing the total route cost [3], and the charge scheduling problem (CSP), which involves determining the best times to charge EVs such that it minimizes the cost and strain on the electrical grid [4]. By tackling these issues concurrently, our approach aims to significantly enhance the overall EV user experience.

It is important to address the integration of charge scheduling into the routing problem with EVs as not only will bolster the ability of the existing infrastructure to handle the increasing adoption of EVs, but also because it could be used to enforce protocols that route traffic among stations to benefit more involved parties than just the drivers themselves. In this paper, we suggest a protocol that minimizes peak loads at charging stations to benefit charging station operators without inconveniencing 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 those stations are, how many chargers are available to use, and the output rate of the available chargers. The decision is left to the driver on which option is the most time-efficient for them and the focus of this selection process is individual convenience. This makes it hard to predict the traffic at each station as it requires the drivers to handle the 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 amongst several charging stations without increasing the average length or distance of each trip by more than 5% compared to the baseline alternatives. 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 the drivers.

This paper is organized as follows: Section 2 describes the problems being solved and the components used by the proposed solution to solve them. Section 3 examines existing approaches to addressing the VRP and CSP. Section 4 describes the methodology for implementing MERL. Section 5 presents the results obtained from the application of MERL. Section 6 concludes the paper, summarizing the findings and their implications as well as how our approach could be improved upon in future research.

II. Background

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

A. Dijkstra's Algorithm

Path-finding algorithms are computational procedures used for finding the shortest path between two points, or nodes, in a weighted graph. These algorithms have a wide range of applications, from routing in transportation networks [5] to protein folding in bioinformatics [6].

Dijkstra's algorithm [7] is a prominent algorithm used in graph theory for finding the shortest paths from a single source node to all other nodes in a graph, which may represent, for example, road networks. It is widely used in several applications including Google Maps for route optimization [8]. The algorithm has been adapted and improved for specific applications such as vehicle path planning [9], mobile robot path planning [10], and obstacle avoidance [11].

The algorithm operates on a weighted graph G and produces a shortest path tree which represents the shortest path from a given source node to every other node in the graph [12]. The algorithm maintains a priority queue of nodes, where each node has a tentative distance value. Initially, all distances are set to infinity, except for the source node which is set to zero. The algorithm then repeatedly extracts the node with the smallest distance from the nodes in the priority queue and updates the distances of its adjacent nodes. If the newly calculated distance to a neighbor is less than its current distance, the algorithm updates the shortest distance and predecessor for that neighbor.

This process continues until all nodes have been visited, resulting in a shortest path tree. The shortest path to any node can then be found by backtracking from the destination node to the source node using the predecessor information [12].

B. Reinforcement Learning

Reinforcement learning (RL) is a sub-field of machine learning where agents learn to make decisions by interacting with an environment [13]. Unlike supervised learning, where the agent is explicitly taught, in RL, the agent learns from the consequences of its actions. These consequences are quantified via 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 the agent can take.
- $P : S \times A \times S \rightarrow [0, 1]$ is the transition probability function, defining 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 the state s.
- $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function that 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, determining the present value of future rewards.

In the traditional usage of RL, the agent's behavior is defined by a policy $\pi : S \times A \rightarrow [0, 1]$, which specifies the probability $\pi(a|s)$ that the 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 [14].

In this paper, we leverage Deep Reinforcement Learning (DRL), a sub-field of RL that uses deep learning techniques to handle complex environments with large state-action spaces. 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 the state s, takes action a, and follows policy π thereafter. The Q-function is instrumental in deriving the policy of the agent, and the accuracy of the Q-function is indicative of the performance of the agent. The optimal Q-function $Q^*(s, a)$ corresponds to the maximum expected return achievable by following any policy, given the agent starts in state s and takes action a.

C. 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 [15]. 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, $\mathcal{E}$ 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 as 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 use of a target network and a replay buffer are two key techniques in DQL. The target network helps to stabilize the learning process by providing a fixed target for the Q-network. The replay buffer allows 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 [16], controlling robotic arms [17], and playing the game of Go [18]. In this paper, we propose to use DQL to enhance the performance of path-finding algorithms.

III. Literature Review

Existing research [19][20][21] which addresses the VRP with EVs is limited in scope and fails to consider important factors such as the current load being placed on a charging station's power supply and the traffic at charging stations. Previous attempts to solve this problem primarily focus on minimizing the time an EV owner spends driving [22][23][24] or minimizing the distance they travel [25][26][27]. Similarly, existing research addressing the CSP with EVs [28][29][30] fails to consider the opportunity to route an EV to a different station entirely to balance the power loads across the 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 [22]. When specifying a route, the Trip Planner will automatically include recommended stops along the path to ensure the driver can arrive at their destination without running out of energy. This solution does not consider factors related to each charging station such as their traffic and the energy load caused by it. In contrast, the application of MERL makes recommendations that include those additional factors to improve the experience for both the driver and charging station operators.

As mentioned by Abid *et al*. [19], the challenge 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 [31] and the scheduling of EV charging is treated as mixed-integer linear programming problems [32]. Artificial intelligence has been used to solve the VRP and CSP separately [28][33] but has not been applied to the combinations of both together.

Keonwoo and Ilkyeong [34] used RL to deal with the scheduling and allocation of charging EVs but did not consider the routing of the EVs to the allotted charging stations. Cai *et al*. [35] used Dijkstra's algorithm with a few performance enhancements to match drivers to charging stations that are close to them. Similar to Tesla's Trip Planner, the main downside of this approach is that it also does not consider the load placed on the charging stations nor the wait times caused by increased traffic at centralized stations. Lee *et al*. [36] used RL to select a charging station for an EV to stop at 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 pre-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 Dijkstra's algorithm to find the shortest path when considering both the traffic at charging stations as well as the distance/time the path takes to travel.

The first section details how RL is used to enhance Dijkstra's algorithm, the second section discusses how the integration is applied 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 approach in this paper uses a new technique 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 the individual factors will be referred to as the factors' impact rating (IR). To find the optimal IR to assign to each factor, an RL algorithm will update them, calculate the edge weights based on the assigned ratings, then give the new weighted graph to a path-finding algorithm.

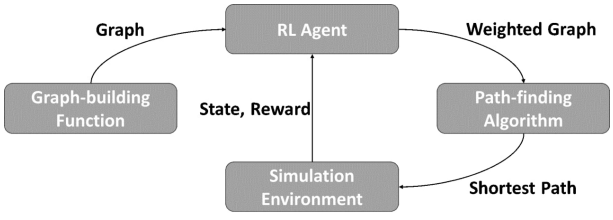

Fig. 1. Feedback loop for MERL.

Figure 1 illustrates the feedback loop which 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 effectively decide the weights of the graph by first deciding on the IRs, then use 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 reward obtained using the given path which the RL agent learns from.

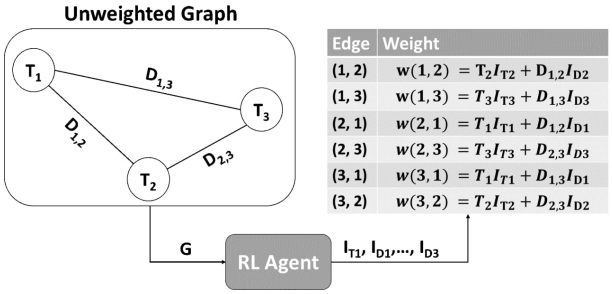

Fig. 2. Using MERL with distances and traffic levels.

Figure 2 shows how MERL converts an unweighted graph into a weighted one in the context of the proposed solution. 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 the traffic level I_{Ti} and distance I_{Di} at node i. It then uses the IRs to get a weighted sum of the traffic level and distance which becomes the edge weight $w(j, i)$ connecting the 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 I_{T2} by the traffic level at node 2 T_2 and adds it to the product of the IR I_{D2} with the distance between the nodes $D_{1,2}$. 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.

Though the distance between the 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 asymptotic 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.

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 figure 3, each output neuron is an IR that takes a value between 0 and 1 which is used to multiply its associated attribute value. The formula to determine the number of outputs from the RL agent is shown in Eq. 4:

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

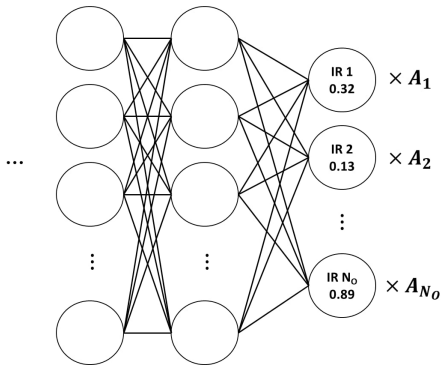

Fig. 3. Output structure of the neural network used in MERL.

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 technique has been applied to the problem of routing EVs while enforcing the constraint that the EVs need to stop at a charging station before going to their destination. The objective function considers multiple factors and aims to minimize both the time spent traveling, as well as the peaks in traffic levels at charging stations.

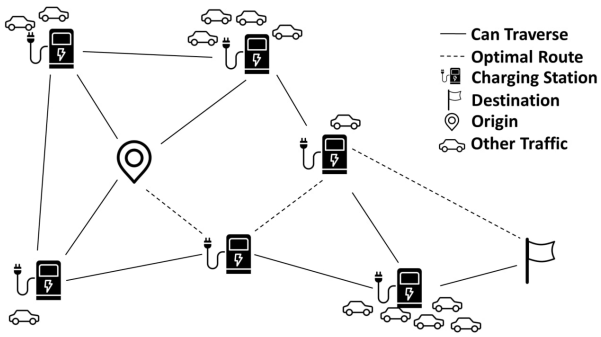

Fig. 4. Simulation environment.

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 has to 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.

Each station has a traffic and distance attribute, meaning the RL agent has to 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 1 value for every node instead of 1 value for every attribute. This works because there are only 2 attributes considered in this application of MERL, so the IR for traffic I_{Tj} assigned to node j can also be used to calculate the IR for the distance I_{Dj} as shown in Eq. 5:

$$I_{Dj} = (1 - I_{Tj}) \tag{5}$$

To ensure 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 technique, 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 get 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 the origin to the 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 not charging at a station, the battery level will decrease proportionately to how far it traveled.

The path given to the environment after using Dijkstra's algorithm is a set of steps consisting of the destination to travel to 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 is the case in other research [37], 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. 6:

$$C_{out}(T, t, c) = \min\left(\frac{C_{ind}}{T_{t,c}}, C_{ind}\right), \tag{6}$$

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, 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 first stopping to charge at least once. When an EV chooses to charge at a station, it increases that station's traffic level by 1 when it arrives then decreases it by 1 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 when finding the path for the first EV, the traffic at each station is 0 allowing it to simply take the route with the shortest distance. Because each EV needs to stop at least once on its path, the second EV will make a decision knowing one station has a traffic level of 1. This continues until the final EV decides on a route knowing where every other EV will stop at 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 starting battery of each EV varies by EV to give a more accurate representation of the real-world environment. However, the starting battery must be low enough to prevent the EV from traveling directly to the destination as 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 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 it is the sum of these individual rewards that is returned to the RL agent after the simulation completes. The reward function used to train the RL agent is shown in Eq. 7:

$$R = -\sum_{t=0}\left(\frac{D_{t,rem}}{D_T} + \max_c T_{t,c} + \sum_{h=0}^{t} M_h\right), \tag{7}$$

where $D_{t,rem}$ is the current distance remaining to the destination at timestep t, D_T is the total distance from the 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 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 the distance traveled.

TABLE I
COMPARISON OF TRAFFIC DURING SIMULATION

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

V. EVALUATION

To evaluate the proposed method, the agent was given control of an environment where 100 EVs had to find the shortest path for their unique and distinct routes which are between 10-20km or 90-180% the distance of London Ontario's estimated radius [38]. To compare the proposed method to alternative baseline approaches, the simulation was run using 4 different cases across 3 seeds.

Case 1 where Dijkstra's algorithm was applied considering only the traffic levels, case 2 where Dijkstra's algorithm was applied considering only the distances, case 3 where Dijkstra's algorithm was applied giving equal consideration to both the traffic level and distances, and case 4 where MERL was applied to find a balance between the impact of both traffic levels and distances. Cases 1-3 are the baselines to compare to the proposed solution, and case 4 is the proposed method itself.

Each EV had the same fixed number of chargers available to it which are based on real-world data. The set of chargers contains 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 [39], along with information pertaining to those charging stations including the geographical coordinates. Each EV has its own starting battery percentage which ranges from 13-15% of the total battery level but is restricted from going below 10%. This is to ensure each EV will be required to stop at a charging station before reaching its destination but gives certain EVs the option of traveling to stations outside of the set near the route origin.

Tables I, II, and III show how the proposed method compared to the various baselines. Note that the values are taken across all 3 seeds. Table 1 compares the performance of the 4 cases at balancing the traffic levels across stations, table 2 compares the travel times of the 4 cases which includes the time spent driving and charging, and table 3 compares the distances traveled by each case.

Table I shows that case 4 using MERL was able to reduce the peak traffic level by 65% compared to case 2 which aimed to minimize distance traveled. Figure 5 also shows that MERL consistently balanced traffic across the various stations better case 2. Case 2 clustered traffic across a small number of stations, which is highlighted by the various orange peaks in figure 5.

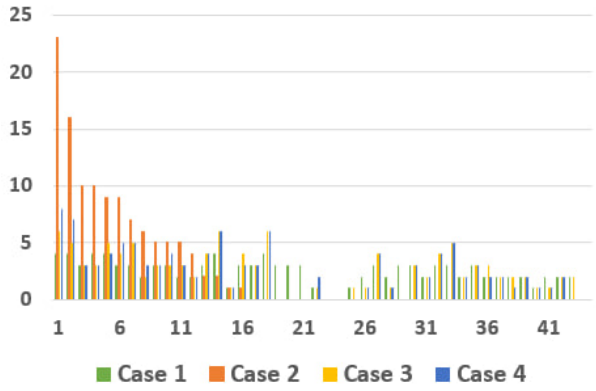

Fig. 5. Peak traffic at each charging station.

TABLE II
COMPARISON OF TIME SPENT TRAVELING IN SIMULATION

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

Table II shows that the number of timesteps taken varies little across all cases, but that case 2 which only aims to minimize distance takes the longest amount of 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 III shows that case 4 using MERL decreased the maximum distance by 74% compared to the solution that only focused on balancing traffic. The proposed solution also performed better than the baseline which evenly balanced focus between traffic and distance by 11%. This shows why although case 1 can effectively balance traffic loads, it is not a viable solution as the routes chosen would be too inconvenient for the driver. Case 4 however, only increased the maximum trip distance by less than 2.5km and the average trip distance by less than half a kilometer compared to case 2 which has the best distance metrics but clusters traffic.

Lastly, figure 6 shows the reward function throughout training the RL agent across the 3 seeds. The solid green line is the average across the 3 seeds, and the area around contains the min to max values. In all three seeds, the training converged to a stable reward value around episode

TABLE III
COMPARISON OF DISTANCE TRAVELED IN SIMULATION

	Avg Distance	Max Distance
Case 1 (Traffic Focused)	26.881	121.700
Case 2 (Distance Focused)	20.330	30.000
Case 3 (Evenly Balanced)	21.228	36.120
Case 4 (MERL Integrated)	20.774	32.470

50. Note that each episode contains 100 EVs and thus 100 learning experiences, so the agents converged after roughly 5000 training experiences.

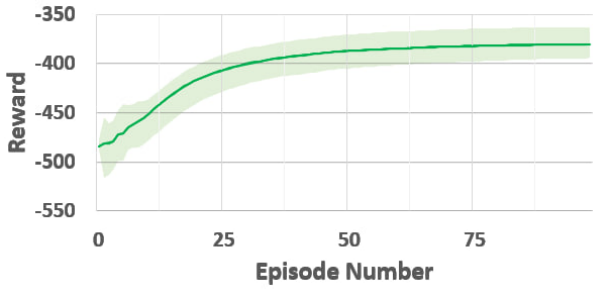

Fig. 6. Reward function during training.

VI. Conclusion

The results of this study indicate that the proposed solution can effectively balance the load across charging stations while also providing a satisfactory user experience for EV owners. The proposed solution 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, we can optimize the charging process for EVs and reduce the output strain on charging stations.

The proposed solution integrates the charging process into the routing process, providing a simple and convenient experience for EV drivers while also reducing average travel times by more than 5% compared to a baseline emulating the current industry standard. By making the charging process more convenient 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.

A perfect solution to the vehicle routing and charge scheduling problems will be subjective to the person routing their trips. Some drivers may prioritize minimizing the travel times, while others may prioritize minimizing the charging cost, or something else entirely. The proposed methodology could address these preferences by updating the reward function in Eq. 7 to give more emphasis to either the distance or traffic level by multiplying the ratios that represent their importance. The reward function can also be updated to consider more factors such as trip cost, resulting in a highly adaptable solution to the VRP and CSP.

Overall, the integration of RL as a solution to the wide field of EV problems has plenty of potential. It's already being used to solve things such as autonomous driving [40], improving battery life [41], and much 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.

While the results of this study are promising, there are several areas 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 that includes factors such as road traffic congestion and variable charging station availability.

The MERL technique was evaluated on only the vehicle routing and charge scheduling problems, but as it is essentially just a way to find weights for a graph when balancing multiple factors, it could theoretically be used in other applications which require weighted graphs. Fields such as networking, robotics, social media, and video games all use path-finding algorithms and weighted graphs, and could thus potentially benefit from the integration of the MERL technique.

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. Dibbelt, J., Strasser, B. & Wagner, D. *Fast exact shortest path and distance queries on road networks with parametrized costs* in *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2015), 1–4.
6. Bonet, B. & Geffner, H. Planning as heuristic search. *Artificial Intelligence* **129,** 5–33 (2001).
7. Javaid, A. Understanding Dijkstra's algorithm. *Available at SSRN 2340905* (2013).
8. Lanning, D. R., Harrell, G. K. & Wang, J. *Dijkstra's algorithm and Google maps* in *Proceedings of the 2014 ACM Southeast Regional Conference* (2014), 1–3.
9. Zhu, D.-D. & Sun, J.-q. A New Algorithm Based on Dijkstra for Vehicle Path Planning Considering Intersection Attribute. *IEEE Access* (2021).
10. Li, F., Du, Y. & Jia, K. Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm. *Scientific Reports* (2021).
11. Alshammrei, S., Boubaker, S. & Kolsi, L. Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance. *Complexity* (2022).
12. Zhao, J., Liu, S. & Li, J. Research and Implementation of Autonomous Navigation for Mobile Robots Based on SLAM Algorithm under ROS. *Sensors* (2022).

13. Wiering, M. A. & Van Otterlo, M. Reinforcement learning. *Adaptation, learning, and optimization* **12,** 729 (2012).
14. Puterman, M. L. Markov decision processes. *Handbooks in operations research and management science* **2,** 331–434 (1990).
15. Mnih, V. *et al.* Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).
16. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *nature* **518,** 529–533 (2015).
17. 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).
18. Silver, D. *et al.* Mastering the game of Go with deep neural networks and tree search. *nature* **529,** 484–489 (2016).
19. Abid, M., Tabaa, M., Chakir, A. & Hachimi, H. Routing and charging of electric vehicles: Literature review. *Energy Reports* **8,** 556–578 (2022).
20. Sweda, T. M., Dolinskaya, I. S. & Klabjan, D. Adaptive routing and recharging policies for electric vehicles. *Transportation Science* **51,** 1326–1348 (2017).
21. Desaulniers, G., Errico, F., Irnich, S. & Schneider, M. Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research* **64,** 1388–1405 (2016).
22. Tesla. Tesla Trip Planner Documentation. https://www.tesla . com / ownersmanual / modelx / en_us / GUID - 01F1A582 - 99D1 - 4933 - B5FB - B2F0203FFE6F . html (2023).
23. Li, J., Wang, F. & He, Y. Electric vehicle routing problem with battery swapping considering energy consumption and carbon emissions. *Sustainability* **12,** 10537 (2020).
24. 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).
25. 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).
26. 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).
27. 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).
28. 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.
29. 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).
30. 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).
31. Little, J. D., Murty, K. G., Sweeney, D. W. & Karel, C. An algorithm for the traveling salesman problem. *Operations research* **11,** 972–989 (1963).
32. Vielma, J. P. Mixed integer linear programming formulation techniques. *Siam Review* **57,** 3–57 (2015).
33. Nazari, M., Oroojlooy, A., Snyder, L. & Takác, M. Reinforcement learning for solving the vehicle routing problem. *Advances in neural information processing systems* **31** (2018).
34. Park, K. & Moon, I. Multi-agent deep reinforcement learning approach for EV charging scheduling in a smart grid. *Applied Energy* **328,** 120111 (2022).
35. 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).
36. 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).
37. 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).
38. 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).
39. Network, N. D. Alternative Fuel Stations API (2023).
40. 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).
41. Harper, G. *et al.* Recycling lithium-ion batteries from electric vehicles. *nature* **575,** 75–86 (2019).