Comparative Analysis of Reinforcement Learning for Reliable Real-Time EV Routing and Charging

Majid Ghasemi
Dept. of Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
mghasemi@wlu.ca

Dariush Ebrahimi
Dept. of Computer Science
Wilfrid Laurier University
Waterloo, Ontario, Canada
debrahimi@wlu.ca

Fadi Alzhouri
Dept. of Elec. & Computer Eng.
GUST University
Mishref, Kuwait
alzhouri.f@gust.edu.kw

***Abstract*—This paper presents a comparative analysis of reinforcement learning (RL) and deep reinforcement learning (DRL) techniques for optimizing reliable electric vehicle (EV) routing and charging in dynamic urban environments. By modeling the EV routing problem as a Markov Decision Process, our study accounts for real-world constraints such as battery levels, charging station availability, traffic conditions, and range anxiety. We evaluate three approaches: traditional Q-Learning, heuristic-augmented Q-Learning (TQL), and Deep Q-Learning (DQL) across simulated networks of varying scales. The experimental results demonstrate that DQL achieves faster convergence and consistently outperforms the other methods in minimizing travel time, improving distance efficiency, and maximizing cumulative rewards while ensuring more reliable route planning. These findings highlight the potential of advanced RL frameworks to enhance EV operational efficiency and charging reliability, offering valuable insights for future smart and sustainable transportation systems.**

I. Introduction

Autonomous Electric Vehicles (AEVs) have emerged as a new and prominent approach and are expected to bring sustainable ways to alleviate traffic congestion, reduce energy consumption, and provide on-demand mobility services [1]. This surging demand, driven by cost-effectiveness, government regulations, reduction of emissions, and technological advancement, emphasizes the need to optimize such systems in a smart urban environment [2]. Despite intensive research on the solution to a large host of ethical, safety, and technical problems with AEV, publications that integrate autonomous charging and driving allow modest optimality in many cases due to complexity.

This paper emphasizes the need to identify the quickest reliable path for an AEV from its source to its destination while ensuring its battery charge remains above a recommended threshold to mitigate range anxiety. It takes into account where charging stations are located and aims to optimize battery usage. Deciding when and where to charge is crucial, but so is determining how much charge is needed to complete the journey. This is especially important since charging an EV still takes more time than refueling with gasoline [3]. To address these challenges, the study proposes automating battery level management, route planning, and selecting the best charging stations based on various factors. The objective is to ensure that the AEV reaches its destination quickly while using its battery efficiently. More precisely, we dive into the dynamic nature of the environment in which AEVs operate, aiming to tackle the challenge of optimizing reliable routing and charging scenarios. Building upon the previous work [4], which explored heuristic methods for EV route optimization with charging considerations, and [5], where authors investigated the usage of Reinforcement Learning (RL) to solve the problem, we recognize the need for methodologies capable of adapting to temporal fluctuations to achieve optimal results. Our study aims to provide advanced RL and Deep Reinforcement Learning (DRL) techniques to improve the results achieved while adhering to the same constraints and objective.

The literature review indicates several existing approaches to EV routing and charging optimization. Zhang et al. [6] proposed a nonlinear programming model considering EV speed and loads for energy-efficient routing, while Hiermann et al. [7] emphasized distance-based approaches. Bruglieri et al. [8] utilized Mixed Integer Linear Programming (MILP) to serve fleets of EVs, and Michael et al. [9] employed tabu search heuristic methods. Machine learning techniques, as demonstrated by Masikos et al. [10] and Cauwer et al. [11], have also been utilized. Furthermore, studies by Bi et al. [12] and Cataldo et al. [13] address charging station coordination and battery charging strategies, respectively.

In the previous work, in [4], authors investigated the heuristic approaches. Later, in [5], they implemented a Q-learning approach and a new heuristic approach. A primary focus of our research is the integration of other RL and DRL algorithms to tackle the dynamic AEV routing problem.

Although Q-Learning performed decently in the previous work, we believe that other algorithms may perform better, and the need for comparison among various algorithms is felt. Thus, we develop other RL and DRL algorithms to compare with the Q-learning, as a baseline, to see the effects of the newly developed algorithms. The first algorithm is TQ-Learning, where we initialized the Q tables with the results derived from the Dynamic TERC2 algorithm [5]. Another algorithm used in this study is Deep Q-Learning (DQL), which

The authors would like to express their sincere gratitude to the GUST Engineering and Applied Innovation Research Center (GEAR) at Gulf University for Science and Technology (GUST) for their support.

uses Neural Networks [14]. We compared the results achieved by these algorithms to give a thorough comparison.

By conducting detailed evaluations across networks of different scales, we demonstrate that our new algorithms significantly surpass traditional Q-Learning in this context. Our study uniquely compares three distinct RL methods, providing researchers with valuable insights into their performance and revealing more effective strategies for practical applications. Moreover, our work contributes to the ongoing advancement of optimizing electric vehicle routing in dynamic environments.

The remainder of this paper is organized as follows. Section II introduces the system model and describes the problem. Section III briefly explains the implemented heuristic method to solve the problem. The Reinforcement Learning approaches are explained in Section IV. Section V presents the performance evaluation, and finally, Section VI concludes the paper and talks about the key findings.

II. System Model and Problem Description

The system model utilizes a weighted graph $G = (N, E)$ to represent the network, where N comprises nodes such as intersections and charging stations, and E consists of road segments connecting these nodes. Each node connection is given values derived from distance, travel duration, and battery status factors. As illustrated in Fig. 1, the starting and end points are shown by black nodes marked as S and D, respectively, whereas intersections and charging facilities are illustrated as blue and red nodes. The electric vehicle (EV) traverses from the source to the destination following the dotted arrow, with its energy depleting at a rate of $E_{\text{Depletion}}$ and replenishing when charging at a rate of E_{Charging}. To enhance battery longevity, the system enforces constraints on the minimum (B_{min}) and maximum (B_{max}) battery levels. Charging stations are identified by the set I_{Station}, and each road segment (i, j) has a fixed traversal time $T^{\text{Edge}}_{(i,j)}$. The primary objective is to determine the optimal route that maintains the battery level within the specified limits while minimizing the total travel time, which includes charging durations. Additionally, the EV must ensure that upon arrival at the destination, the battery level B_D satisfies $B_D = B_{\text{min}} + E_{\text{toStation}}$, guaranteeing sufficient charge to reach the nearest charging station.

The routing challenge becomes even more intricate due to the ever-changing nature of the surroundings. Factors such as inconsistent waiting times at charging stations, varying traffic speeds, and unpredictable battery consumption rates all contribute to the complexity of the problem. For instance, an EV starting with a battery level of $B_S = 60\%$ may need to divert to a charging station to prevent the battery from dropping below the threshold $B_{\text{min}} + E_{\text{toStation}}$, as demonstrated in the journey from node 1 to node 5 for recharging. This dynamic setting classifies the routing problem as NP-hard, analogous to the Traveling Salesman Problem (TSP). Unlike the classical TSP, the EV is not required to visit every charging station but must strategically select stops based on the necessity to maintain adequate charge levels. This added flexibility introduces additional complexity, necessitating the development of

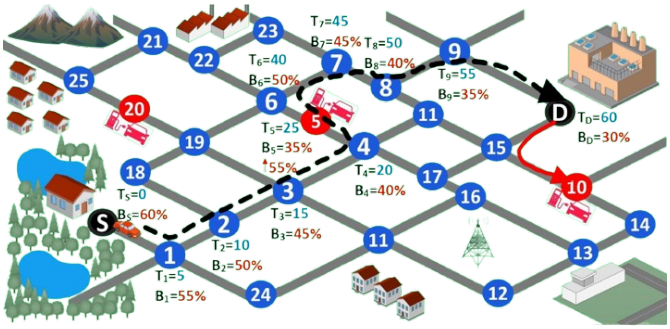


Fig. 1: A visual representation of the system framework is provided, where intersections appear as blue nodes, and charging stations are marked in red. The source and destination points are denoted by S and D, respectively. Roads linking these nodes are depicted as edges. A dashed black arrow indicates the movement direction of an EV along a road, while each node is associated with the EV's estimated arrival time (T_i) and corresponding battery percentage level (B_i).

advanced and innovative strategies to effectively address the routing and charging challenges. In subsequent sections, we will explore practical solutions tailored to these dynamic and state-dependent aspects of EV routing.

III. The Heuristic Approach: Dynamic TERC2

Alg. 1 details the dynamic TERC2 approach, which largely follows the TERC2 method described in [4], except for the modification applied when selecting the next charging station under conditions of insufficient battery capacity. The algorithm accepts a graph $G(N, E)$, the initial battery level B_s, the source node S, and the destination node D as inputs, and it computes the fastest route from S to D while addressing the constraints necessary for preserving battery life in a dynamic environment. Notably, line 3 of the algorithm updates edge-related information, such as traffic speed and battery consumption rate, thereby allowing the system to adjust its route decisions in response to changing conditions. This real-time update mechanism enhances the algorithm's ability to optimize route selection, particularly when determining the nearest charging station from either the source or the destination.

Furthermore, the dynamic TERC2 method refines its process by computing a temporary time T_{temp} at line 17, which sums the time required to reach a charging station from the source (T_{temp1}) and the time from the charging station to the destination (T_{temp2}). Lines 18 to 20 then collectively identify the charging station that minimizes the overall travel time between the source and the destination. An important benefit of this approach is its capability to reroute the electric vehicle away from charging stations or destination points that initially appeared optimal but later proved inefficient due to evolving traffic conditions.

IV. Reinforcement Learning Approaches

Reinforcement Learning (RL) involves an agent interacting with its environment to develop a strategy that maximizes

Algorithm 1: Dynamic TERC2

Input: Graph $G(N, E)$, B_s, S, D.
Output: P_T: The fastest path in Dynamic Environment by considering all constraints from S to D.

$T_D = 0$;
Function `FindPath`(S, D, B_S)**:**
  $E \longleftarrow E_{state}$;
  $P_T \longleftarrow Dijkstra(S, D)$;
  $T_P \longleftarrow TimeOnPath(P_T)$;
  $B_{used} \longleftarrow BatteryUsedOnPath(P_T)$;
  if $B_i - B_{used} >= B_{min} + E_{toStation}$ **then**
    $T_D \leftarrow T_D + T_P$;
    return P_T
  else
    $T_P = L$;
    for *each* $i \in \mathcal{I}_{Station}$ **do**
      $P_{temp1} \leftarrow Dijkstra(S, i)$;
      $P_{temp2} \leftarrow Dijkstra(i, D)$;
      $T_{temp1} \longleftarrow TimeOnPath(P_{temp1})$;
      $T_{temp2} \longleftarrow TimeOnPath(P_{temp2})$;
      $T_{temp} = T_{temp1} + T_{temp2}$;
      if $T_P > T_{temp}$ **then**
        $T_P = T_{temp}$;
        $S_{new} \leftarrow i$;
    $B_{new} = B_{max}$;
    $T_D = T_D + T_{temp1} + G_i$;
    return $P_{temp1} + FindPath(S_{new}, D, B_{new})$

long-term rewards [15]. In our work, we frame the problem as a Markov Decision Process (MDP) defined by the tuple (S, A, γ, P, R), where the set of states (S) represents a finite collection of conditions, with each state s_t corresponding to a specific time step t. The actions (A) available are the possible decisions the agent can take at any given state s_t, with each action denoted as $a_t \in A$. The discount factor (γ) is a parameter within the interval $[0, 1)$ that determines the relative importance of future rewards compared to immediate ones. The transition function (P), expressed as $P(s_{t+1} \mid s_t, a_t)$, specifies the probability of moving from the current state s_t to the next state s_{t+1} after taking action a_t. Lastly, the reward function (R), denoted as $R(s_t, a_t)$, assigns an immediate reward for executing action a_t in state s_t at time t.

Agent: Here, the agent is the EV itself. The EV is responsible for making decisions at every stage of its journey, choosing actions that help it navigate a changing environment while aiming to reduce its overall travel time.

State: The state space captures all the possible conditions the EV might encounter. Each state is characterized by several attributes: the EV's location within the road network (depicted by nodes), its current battery level, and the elapsed travel time. Additionally, reaching the destination is treated as entering a terminal state, which signals the end of the journey.

Actions: The available actions primarily involve moving the EV from one node to an adjacent one. Additionally, when the EV finds itself at a charging station, it has the extra option to recharge its battery to certain level(s).

Reward: The reward function is tailored to the goal of minimizing travel time. It assigns a negative value equal to the time consumed, which includes both the travel time between nodes and the time spent charging. Charging time is computed based on the difference between an optimal battery level (set at 80%) and the current battery level, expressed as

$$reward_{charging} = -(80 - current_{battery}).$$

Moreover, if the battery level falls below 20%, a significant penalty is imposed to encourage practices that extend battery longevity. Consequently, charging is limited to a maximum of 80%. By modeling our problem as an MDP, the EV, acting as the agent, learns to chart an optimal path through its environment. Through selecting actions that transition it to more advantageous states, the EV minimizes travel time while maximizing cumulative rewards. The details of the proposed RL and DRL approaches are presented below.

A. Q-Learning

The first used algorithm in this paper is Q-Learning, firstly introduced by [16], which is a model-free algorithm. Alg. 2 begins by initializing the Q-table for all state-action pairs to zero (line 1). This Q-table is fundamental to the Q-learning process as it stores the estimated rewards for executing specific actions from each state [15]. The algorithm is designed to work on a graph $G(N, E)$ with a defined start state S, a destination D, and designated charging stations C. Such a setup tailors the learning process to a constrained path planning problem, taking into account operational constraints such as battery charge and the availability of charging facilities.

For each of the K episodes (line 2), the algorithm sets the current node to the start state S (line 3) and records the initial path. Within each episode, a while-loop is executed until a termination condition is satisfied. At each iteration, an action is selected using an ϵ-greedy policy based on the current Q-table (line 6), which provides a balance between exploring new actions and exploiting known high-reward actions. The chosen action leads to the next state N_{next} (line 7), and the path is updated accordingly (line 8). A central aspect of this approach is the computation of the temporal difference error, δ, given by

$$\delta \leftarrow R + \gamma \times \max_{a'} Q(s', a') - Q(s, a)$$

(line 9). This error is then used to update the Q-value for the corresponding state-action pair using the formula

$$Q(s, a) \leftarrow Q(s, a) + \alpha \times \delta$$

The algorithm tracks the maximum Q-value change for convergence (line 10) and updates the current node to N_{next} (line 11). If the destination is reached or if the battery charge falls below zero (line 12), and the current reward exceeds the best recorded reward, the algorithm updates the best epoch results (lines 13-15) before exiting the loop. Finally, at the end of each episode,

Algorithm 2: Q-Learning Algorithm

Input: Graph $G(N, E)$, Start State S, End State D, Charging Stations C

Output: Optimal Path P_T from S to D considering all constraints

1 $Q(s, a) \leftarrow 0$ for all state-action pairs
2 **for** $k \leftarrow 1{:}K$ **do**
3 $N_{current} \leftarrow S$
4 $P \leftarrow [N_{current}]$
5 **while** *True* **do**
6 Choose action a using ϵ-greedy policy based on Q
7 N_{next} based on action a
8 $P \leftarrow P + N_{next}$
9 $\delta \leftarrow R + \gamma \times \max_{a'} Q(s', a') - Q(s, a)$ Update Q-table:

$$Q(s, a) \leftarrow Q(s, a) + \alpha \times \delta$$

10 Track maximum Q-value change for convergence
11 $N_{current} \leftarrow N_{next}$
12 **if** $N_{current} = D$ ***or*** *battery charge* ≤ 0 **then**
13 **if** *current_reward* $>$ *best_reward* **then**
14 Update best epoch results with current path and metrics
15 **break**
16 Calculate and record epoch metrics: distance & travel time
17 Decay ϵ and α towards their minimum values
18 **Return** Best epoch results containing optimal path and metrics;

the algorithm calculates and records the epoch metrics, such as distance traveled and travel time (line 16). Additionally, the values of ϵ and α are decayed toward their minimum values (line 17) to gradually shift the behavior from exploration to exploitation. Once all K episodes have been executed, the algorithm returns the best epoch results containing the optimal path and its associated metrics (line 18). This method ensures that the final output is not only the most efficient path in terms of distance and time but also respects the constraints related to battery charge and charging station placement.

B. TQ-Learning (TQL)

As some heuristic methods are implemented to solve this problem, we utilized the best one, Dynamic TERC2, to initialize the Q-tables of the Q-learning algorithm with its results. This leads to faster convergence, as it gives a good sub-optimal place for the algorithm instead of naive zeros, which has little advantage [17].

C. Deep Q-Learning (Deep Q-Networks)

The next proposed algorithm, shown in Alg. 3, merges Q-learning with Deep Learning's Neural Networks to learn control policies directly from the given input [14]. The algorithm begins by initializing the key components necessary for deep Q-learning, as shown in lines 1-4. In these initial steps, the policy network Q_θ and the target network Q_{θ^-} are both set up with random weights, and a replay buffer $\mathcal{D}$ with a fixed capacity is created to store state transitions. An initial battery charge is also assigned a value of 80, and essential hyperparameters, such as the exploration rate (ϵ), discount factor (γ), learning rate (α), and their decay rates, are established. These preparations ensure that the model has the structural and parameter foundations needed to learn the optimal policy for navigating the graph $G(N, E)$ while considering battery constraints and charging stations.

In the subsequent phase, encapsulated within the loop beginning at line 5, the algorithm iterates over a predetermined number of episodes K. At the start of each episode, the initial state S, battery level B, and path list are reset (lines 6-9). The core of the learning process is embedded in a while-loop (line 10) that continues until a terminal condition is met, either by reaching the destination D or by depleting the battery. Within each iteration, an action is chosen using an ϵ-greedy strategy: with probability ϵ a random valid action is selected, while with probability $1-\epsilon$ the algorithm picks the action that maximizes the current Q-value from the policy network (lines 10-12). After executing the chosen action, the algorithm observes the resulting reward, the next state, and the updated battery charge, then stores this transition in the replay buffer (lines 13-14).

When the replay buffer has accumulated a sufficient number of samples (line 18), the algorithm performs a mini-batch update in line 19. For each sampled transition, it computes a target Q-value using the target network Q_{θ^-} by applying the Bellman equation, specifically, taking the maximum predicted Q-value for the next state and battery level, discounting it by γ, and considering whether the state is terminal (lines 20–22). The loss is then calculated as the mean squared error between these target Q-values and the Q-values predicted by the policy network (lines 23-24), and a gradient descent step is executed to update the network parameters accordingly (line 25). Additionally, to maintain training stability, the target network is periodically updated with the latest parameters from the policy network every C steps (line 26).

Finally, the algorithm checks for episode termination conditions within the while-loop. If the destination is reached or the battery is exhausted, the episode is terminated (line 25). Moreover, if the current reward surpasses the best recorded reward, the algorithm updates its best epoch results with the current path and associated metrics (lines 27-29). At the end of each episode, the best total reward, distance, and travel time are recorded, and the exploration rate ϵ is decayed (lines 29-30). Ultimately, after iterating through all episodes, the algorithm returns the best epoch results, which include the optimal path and performance metrics, thereby effectively addressing the constrained path planning problem through deep Q-learning (line 31).

Algorithm 3: Deep Q-Learning Algorithm

Input: Graph $G(N, E)$, Start State S, End State D, Charging Stations C

Output: Optimal Path P_T from S to D

Initialize policy network Q_θ and target network Q_{θ^-} with random weights θ and θ^-;
Initialize replay buffer $\mathcal{D}$ with capacity N;
initial battery charge $B \leftarrow 80$;
Set ϵ, γ, α, and decay rates;
for $k : K$ **do**
  $s \leftarrow S$;
  $battery \leftarrow B$;
  $path \leftarrow [s]$;
  $done \leftarrow$ **False**;
  while ***not*** $done$ **do**
    With probability ϵ select a random action a;
    Otherwise, select $a \leftarrow \arg\max_{a'} Q_\theta(s, battery, a')$ over valid actions;
    Execute action a, observe reward r, next state s', and next battery charge $battery'$;
    Store transition $(s, a, r, s', done, battery, battery')$ in $\mathcal{D}$;
    $s \leftarrow s'$;
    $battery \leftarrow battery'$;
    $path \leftarrow path + [s]$;
    if *enough samples in replay buffer* $\mathcal{D}$ **then**
      Sample minibatch of transitions from $\mathcal{D}$;
      foreach *transition* $(s_i, a_i, r_i, s'_i, done_i, battery_i, battery'_i)$ *in minibatch* **do**
        Compute target Q-value:

$$y_i = r_i + \gamma \times \max_{a'} Q_{\theta^-}(s'_i, battery'_i, a') \times (1 - done_i)$$

      Compute loss:

$$L = \frac{1}{|\text{batch}|} \sum_i \left(y_i - Q_\theta(s_i, battery_i, a_i)\right)^2$$

      Perform gradient descent step on L to update θ;
    Every C steps, update target network: $\theta^- \leftarrow \theta$;
    if $s = D$ ***or*** $battery \leq 0$ **then**
      $done \leftarrow$ **True**;
      if *current_reward* $>$ *best_reward* **then**
        Update best epoch results with current path and metrics;
  Calculate and record the best total reward, distance, & travel time;
  Decay ϵ towards its minimum value;
return Best epoch results containing optimal path;

V. Performance Evaluation

To achieve better results compared to previous work [4,5] for large graphs, in this section, we compare the performance of the implemented methods: TQL, Q-Learning, and DQL.

Our evaluation centers on dynamic environments, where we measure performance in terms of overall travel time, total distance traveled, accumulated rewards, and changes in Q-values. To understand the impact on performance, we vary key parameters such as the graph size, the number of charging stations, and the distance between the source and destination. Furthermore, to account for battery life in our proposed methods, we conducted empirical tests using a maximum battery level of $B_{max} = 80\%$ and a minimum battery level of $B_{min} = 20\%$. Additionally, the EV begins its journey with an initial charge of 80%.

In our implementation for all the algorithms, we used exponential decay for learning rate (α) and exploration rate (ϵ) for $\epsilon-$greedy approach. Exponential decay can be represented as follows:

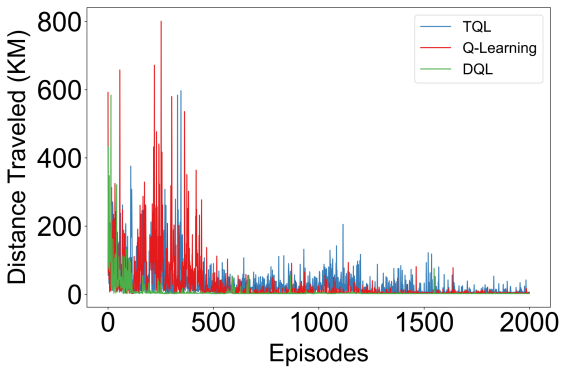
$$\theta_{t+1} = max(\theta_{min}, \theta_t \times decay\_rate) \tag{1}$$

where θ can be the learning rate (α) or exploration rate (ϵ), θ_t is the parameter value at epoch t. $decay\_rate$ is the constant multiplier set to (0.999), and θ_{min} is the lower bound to prevent the parameter from decaying indefinitely. We started with $\alpha = 0.1$, and $\epsilon = 0.25$. The lower bound (minimum) for α and ϵ is 0.001 to ensure the agent does not explore after an adequate number of explorations at first. Other key parameters in our implementation are the discount factor γ, which is set to 0.9, to encourage the agent to value long-term reward as the goal is to minimize the travel time from the source to the destination (considering the longevity of the battery as well). Also, the number of episodes K is set to be equal to 2000 to ensure sufficient training time to achieve the best result.

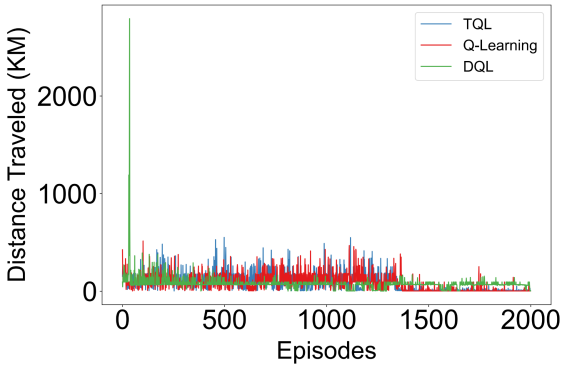
A. Graph Generation & Charging Station Placement

To evaluate the performance of our proposed methods, we generate graphs based on real-world maps, varying in size and comprising nodes connected by weighted edges. To create graphs with a specific number of nodes, we extract a subset from a larger generated graph that meets the desired node count. Given the current range of EV batteries, an electric vehicle can typically travel from any starting point to a destination within a city without requiring a recharge. However, for our experiments, we extend the distances between edges to evaluate how effectively our algorithms can identify charging stations along the route.

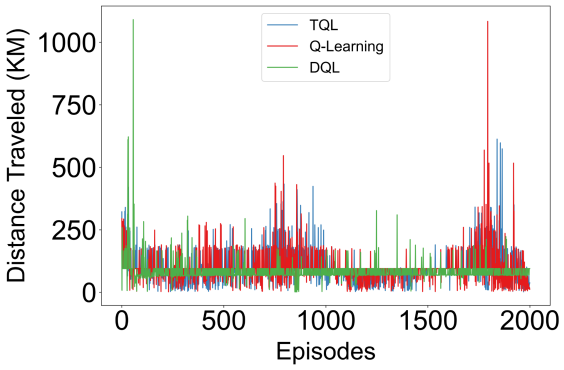
In addition, we attribute three unique pairs of traffic speed and battery consumption values to each edge based on its length. This configuration enables us to mimic varying traffic conditions and dynamically adjust the EV's speed and battery usage at every time step. We also designate specific nodes as charging stations, ensuring that every node is within reach of at least one charging station, even under extreme conditions of battery consumption. This approach guarantees that each algorithm we test always has at least one viable solution.



(a)

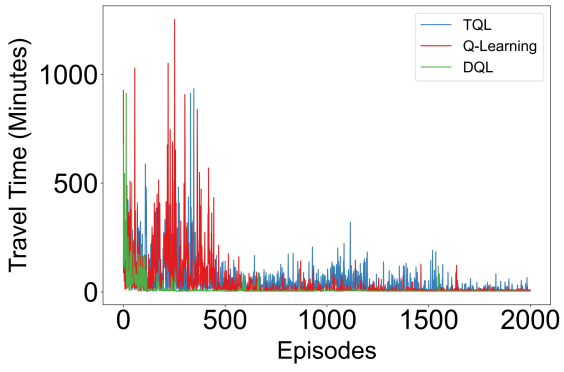


(b)

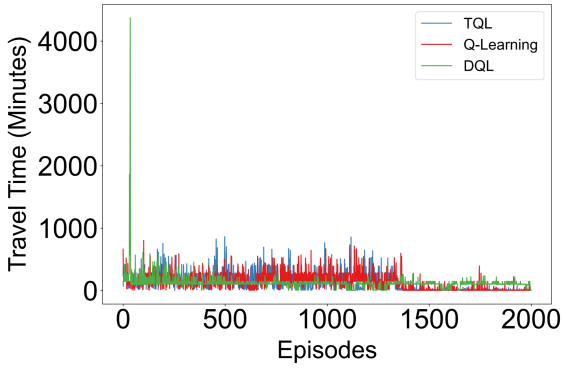


(c)

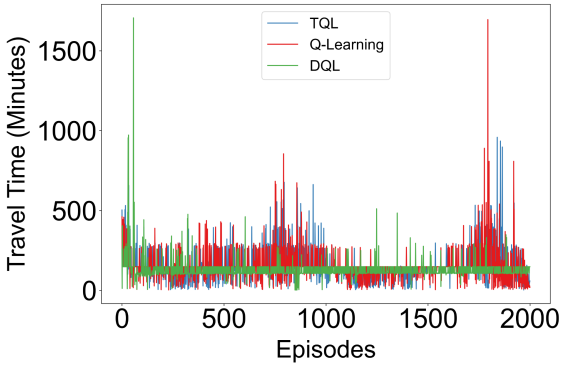
Fig. 2: Comparing the travelled distance (in KM) of different methods (TQL, Q-Learning, and DQL): (a) 125 nodes, (b): 250 nodes, and (c): 500 nodes



(a)



(b)



(c)

Fig. 3: Comparing the travelled time (in Minutes) of different methods (TQL, Q-Learning, and DQL): (a) 125 nodes, (b): 250 nodes, and (c): 500 nodes

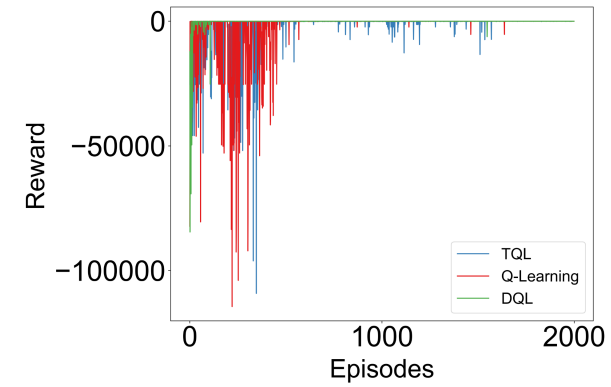
Our method for positioning charging stations is systematic. We start by randomly choosing a node in the graph to act as the initial charging station and add it to the list I_{Station}. For every subsequent charging station, we select the node that is the furthest away in terms of battery consumption from the most recently chosen station and that can be reached with the maximum battery charge B_{Max}. This recursive process continues until the entire graph is covered, ensuring that every area is accessible without depleting the battery.

Finally, we randomly pick a source and a destination node within a predetermined distance range for the EV's journey. For simplicity, we classify each road into one of three traffic conditions, which enables us to simulate and assess the impact of varying traffic statuses on the EV's trip.

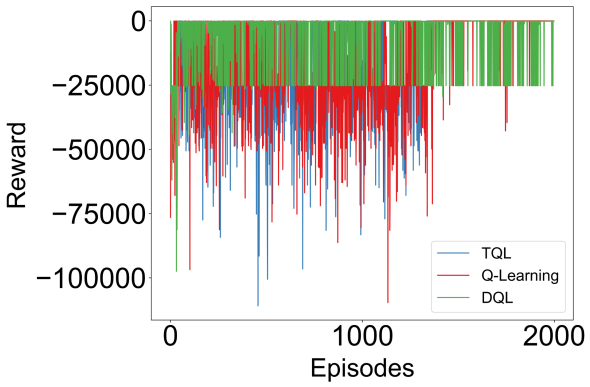
B. Performance Evaluation over Different Networks

To evaluate the proposed methods, we tested them across three different network configurations, each containing 125, 250, or 500 nodes. These variations allowed us to assess performance across diverse network scales.

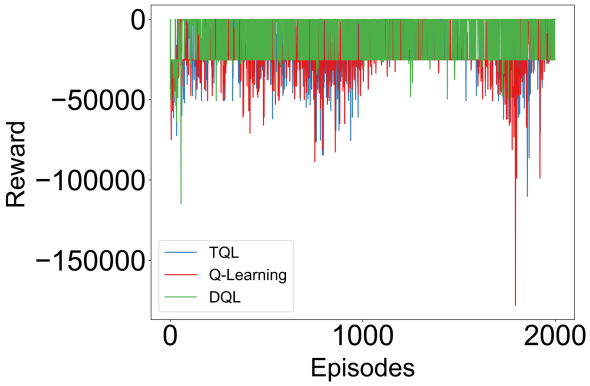
For each network, we report four main metrics. Travelled distance (in kilometers) and time (in minutes), cumulative



(a)

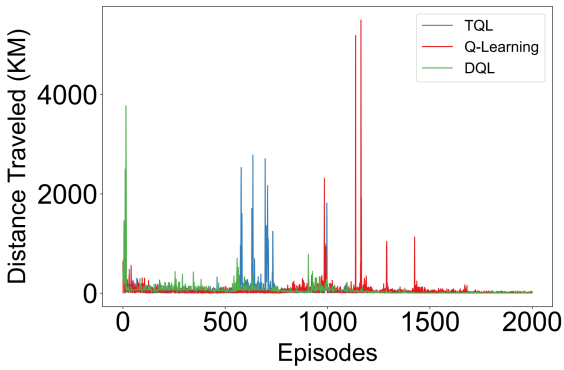


(b)

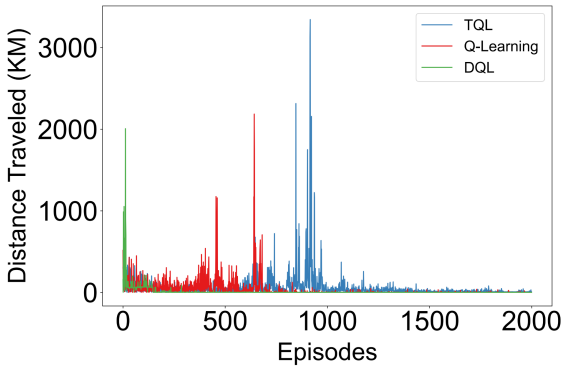


(c)

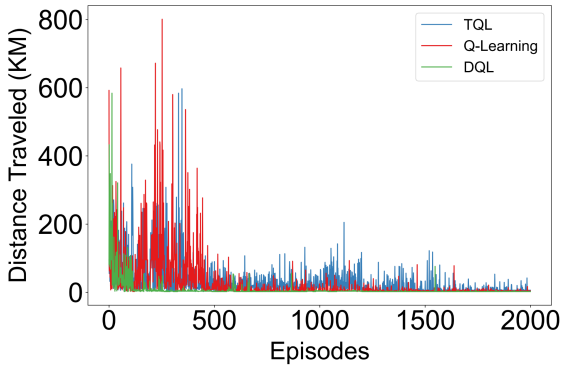
Fig. 4: Comparing the achieved cumulative reward of different methods (TQL, Q-Learning, and DQL): (a) 125 nodes, (b): 250 nodes, and (c): 500 nodes



(a)



(b)



(c)

Fig. 5: Comparing the traveled distance of different methods (TQL, Q-Learning, and DQL) in a network of 125 nodes: (a) radius=30, (b): radius=45, and (c): radius=60 (baseline)

reward that the agent received over all the episodes, and Q-values changes. Analyzing these metrics will give us a detailed perspective of the functionality of each algorithm, the effectiveness of each, and also the convergence.

Then, for the network with $N = 125$, we will examine the effects of changing the placement and the number of charging stations on two main metrics, travelled distance and time.

a) Travelled Distance: Fig. 2 presents a detailed comparison of travel time across three different network configurations. As shown, DQL consistently outperforms the other algorithms and converges faster. However, in the case of 500 nodes, DQL initially struggles in the early episodes, unlike the smaller networks, where it quickly finds the optimal solution. Q-Learning, on the other hand, only surpasses TQL in the $N = 125$ scenario, while in larger networks, it struggles to converge until the final episodes. TQL generally performs as well as or better than Q-Learning, benefiting from its initialization with the heuristic-based Dynamic TERC2 results. In the $N = 125$ case, TQL exhibits noisy behavior due to over-exploitation of suboptimal results from Dynamic TERC2. However, as it explores alternative actions, it achieves varied but non-optimal outcomes.

The same trend can be seen in Fig. 3 as traveling time and distance have a linear correlation with each other. In our case, travel time between each node is a linear combination of the distance, which is calculated as follows:

$$distance = G[N][N+1] \tag{2}$$

where N and N+1 are the adjacent nodes. Travel time is then calculated based on two factors that were derived from maps that were used in the real world: base speed and traffic factor.

$$speed = \frac{base\_speed}{traffic\_factor} \tag{3}$$

$$travel\_time = \frac{distance}{speed} \tag{4}$$

b) Cumulative Reward: Fig. 4 illustrates the cumulative reward for each method. DQL consistently achieved fewer negative rewards across all networks, ultimately attaining the optimal negative cumulative reward. Similar to travel time and distance, TQL outperformed Q-Learning in networks with 250 and 500 nodes. It is important to note that the y-axis in Fig. 4(c) is scaled by 10^6.

As with travel time and distance, DQL converged rapidly, making it the most effective approach for this problem, as minimizing negative rewards is one of the key objectives.

Furthermore, all methods successfully identified the best path for the given instances and converged to the optimal policy. However, DQL reached the optimal policy (π^*) faster than TQL, while Q-Learning demonstrated the weakest performance.

c) Altering the Number of Charging Stations: As discussed earlier, the location and number of charging stations follow a recursive systematic order. To determine whether to place a charging station at a given node, we analyzed the effect of the accessibility radius on travel distance. Increasing this radius results in fewer charging stations across the map. Fig. 5 illustrates these effects. For instance, with a radius of 60, resulting in a sparse distribution of charging stations, all methods required more time to find the optimal policy due to charging constraints. This finding suggests that maintaining an adequate number of charging stations is crucial in real-world scenarios. To ensure a more realistic setup aligned with real-world conditions, we set the baseline number of charging stations using a radius of 60.

VI. Conclusion

This study explores the potential of advanced reinforcement learning techniques to optimize EV routing and charging in dynamic urban environments. Our comparative analysis shows that while heuristic-enhanced Q-Learning accelerates convergence and improves initial performance over traditional Q-Learning, Deep Q-Learning (DQL) demonstrates superior robustness and efficiency in real-time decision-making. DQL consistently achieves lower travel times, reduced energy consumption, and improved overall performance, effectively balancing route optimization and battery management under varying traffic and environmental conditions. These findings highlight the promise of deep reinforcement learning for smart transportation and its relevance in designing future EV systems that are both sustainable and adaptable to real-world challenges. Future research should focus on integrating heterogeneous data sources and adaptive learning mechanisms to bridge the gap between simulations and large-scale practical deployments.

References

[1] L. Zhang, Z. Liu, L. Yu, K. Fang, B. Yao, and B. Yu, "Routing optimization of shared autonomous electric vehicles under uncertain travel time and uncertain service time," *Transportation Research Part E: Logistics and Transportation Review*, vol. 157, p. 102548, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1366554521003069

[2] J. Morris, D. Hone, M. Haigh, A. Sokolov, and S. Paltsev, "Future energy: in search of a scenario reflecting current and future pressures and trends," *Environmental Economics and Policy Studies*, vol. 25, 2022.

[3] M. N. Boukoberine, T. Donateo, and M. Benbouzid, "Optimized energy management strategy for hybrid fuel cell powered drones in persistent missions using real flight test data," *IEEE Transactions on Energy Conversion*, vol. 37, no. 3, pp. 2080–2091, 2022.

[4] D. Ebrahimi, S. Kashefi Mofrad, M. Helae, S. Mohammed, and F. Alzhouri, "Fastest route and charging optimization of an electric vehicle with battery's life consideration," in *Proceedings of the Int'l ACM Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications*, 2023, pp. 39–46.

[5] D. Ebrahimi, S. Kashefi, T. E. A. de Oliveira, and F. Alzhouri, "Efficient routing and charging strategy for electric vehicles considering battery life: A reinforcement learning approach," in *2024 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*. IEEE, 2024, pp. 429–435.

[6] S. Zhang, Y. Gajpal, S. Appadoo, and M. Abdulkader, "Electric vehicle routing problem with recharging stations for minimizing energy consumption," *International Journal of Production Economics*, vol. 203, pp. 404–413, 2018.

[7] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl, "The electric fleet size and mix vehicle routing problem with time windows and recharging stations," *European Journal of Operational Research*, vol. 252, pp. 995–1018, 2016.

[8] M. Bruglieri, M. Paolucci, and O. Pisacane, "A matheuristic for the electric vehicle routing problem with time windows and a realistic energy consumption model," *Computers & Operations Research*, vol. 157, 2023.

[9] M. Schneider, A. Stenger, and D. Goeke, "The electric vehicle-routing problem with time windows and recharging stations," *Transportation Science*, vol. 48, pp. 500–520, 2014.

[10] M. Masikos, K. Demestichas, E. Adamopoulou, and M. Theologou, "Machine-learning methodology for energy efficient routing," *IET Intelligent Transport Systems*, vol. 8, pp. 255–265, 2014.

[11] C. De Cauwer, W. Verbeke, J. Van Mierlo, and T. Coosemans, "A model for range estimation and energy-efficient routing of electric vehicles in real-world conditions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, pp. 2787–2800, 2020.

[12] X. Bi, A. J. Chipperfield, and W. K. S. Tang, "Coordinating electric vehicle flow distribution and charger allocation by joint optimization," *IEEE Transactions on Industrial Informatics*, vol. 17, pp. 8112–8121, 2021.

[13] C. Cataldo-Díaz, R. Linfati, and J. W. Escobar, "Mathematical model for the electric vehicle routing problem considering the state of charge of the batteries," *Sustainability*, vol. 14, 2022.

[14] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[15] M. Ghasemi, A. H. Moosavi, I. Sorkhoh, A. Agrawal, F. Alzhouri, and D. Ebrahimi, "An introduction to reinforcement learning: Fundamental concepts and practical applications," *arXiv preprint arXiv:2408.07712*, 2024.

[16] C. J. C. H. Watkins, "Learning from delayed rewards," 1989.

[17] R. S. Sutton, "Reinforcement learning: An introduction," *A Bradford Book*, 2018.