

Article

SoC-VRP: A Deep-Reinforcement-Learning-Based Vehicle Route Planning Mechanism for Service-Oriented Cooperative ITS

Boyuan Hou ¹, Kailong Zhang ^{1,*}, Zu Gong ¹, Qiugang Li ², Junle Zhou ¹, Jiahao Zhang ² and Arnaud de La Fortelle ³

¹ School of Software, Northwestern Polytechnical University, Xi'an 710129, China; hby1998@mail.nwpu.edu.cn (B.H.); gongzu@mail.nwpu.edu.cn (Z.G.); junlezhou@mail.nwpu.edu.cn (J.Z.)

² School of Computer, Northwestern Polytechnical University, Xi'an 710129, China; qiugang@mail.nwpu.edu.cn (Q.L.); zhangjiahao@mail.nwpu.edu.cn (J.Z.)

³ Centre of Robots, MINES-Paris Sciences et Lettres University, 75006 Paris, France; arnaud.de_la_fortelle@minesparis.psl.eu

* Correspondence: kl.zhang@nwpu.edu.cn

Abstract: With the rapid development of emerging information technology and its increasing integration with transportation systems, the Intelligent Transportation System (ITS) is entering a new phase, called Cooperative ITS (C-ITS). It offers promising solutions to numerous challenges in traditional transportation systems, among which the Vehicle Routing Problem (VRP) is a significant concern addressed in this work. Considering the varying urgency levels of different vehicles and their different traveling constraints in the Service-oriented Cooperative ITS (SoC-ITS) framework studied in our previous research, the Service-oriented Cooperative Vehicle Routing Problem (SoC-VRP) is firstly analyzed, in which cooperative planning and vehicle urgency degrees are two vital factors. After examining the characteristics of both VRP and SoC-VRP, a Deep Reinforcement Learning (DRL)-based prioritized route planning mechanism is proposed. Specifically, we establish a deep reinforcement learning model with Rainbow DQN and devise a prioritized successive decision-making route planning method for SoC-ITS, where vehicle urgency degrees are mapped to three priorities: High for emergency vehicles, Medium for shuttle buses, and Low for the rest. All proposed models and methods are implemented, trained using various scenarios on typical road networks, and verified with SUMO-based scenes. Experimental results demonstrate the effectiveness of this hybrid prioritized route planning mechanism.

Keywords: Service-oriented C-ITS (Soc-ITS); Service-oriented Cooperative VRP (SoC-VRP); vehicle priority; Qualities of Traveling (QoT); deep reinforcement learning; SUMO



Citation: Hou, B.; Zhang, K.; Gong, Z.; Li, Q.; Zhou, J.; Zhang, J.; de La Fortelle, A. SoC-VRP: A Deep-Reinforcement-Learning-Based Vehicle Route Planning Mechanism for Service-Oriented Cooperative ITS. *Electronics* **2023**, *12*, 4191. <https://doi.org/10.3390/electronics12204191>

Academic Editor: Martin Reisslein

Received: 6 September 2023

Revised: 6 October 2023

Accepted: 7 October 2023

Published: 10 October 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Under the support of new information technology, the transportation system has become more intelligent, incorporating powerful capabilities such as deep perception of the traffic environment, autonomous driving, V2X communication, and coordination. With the rapid evolution and deployment of these technologies, it is transitioning from an Intelligent Transportation System (ITS) to a new stage known as Cooperative ITS (C-ITS). In the context of this emerging trend, the prevailing challenges pertaining to traffic congestion, blocked emergency vehicles, and energy waste in conventional transportation systems are progressively being addressed with increasingly intelligent solutions. This has garnered considerable attention and generated significant interest within both the academic and industrial fields.

As is well known, the Vehicle Routing Problem (VRP) [1,2] is one of such key issues which exerts a profound impact on transportation systems. Basically, such route planning

problems for a single vehicles can always be equated to a special Traveling Salesman Problem (TSP) [3] with prioritizing constraints, covering the shortest distance, highways and minimal emissions, etc. However, this problem will become more complicated when planning routes for multiple vehicles in a traffic network, especially if these vehicles have different service attributes, for instance, emergency rescue, mass transit, and private use, as shown in Figure 1. In this particular context, the resolution of such new VRP necessitates the consideration of not only the constraints imposed on individual vehicles but also the interdependencies arising from multivehicle route planning and the distinctive driving needs associated with various service types of vehicles. It is obvious that the coordinated and service-oriented features of such novel VRP pose inevitably fresh challenges to conventional solution approaches. Fortunately, the ongoing advancements in information technology and Artificial Intelligence (AI) enable increasingly efficient resolutions for this problem and have turned it into a research hotspot recently.

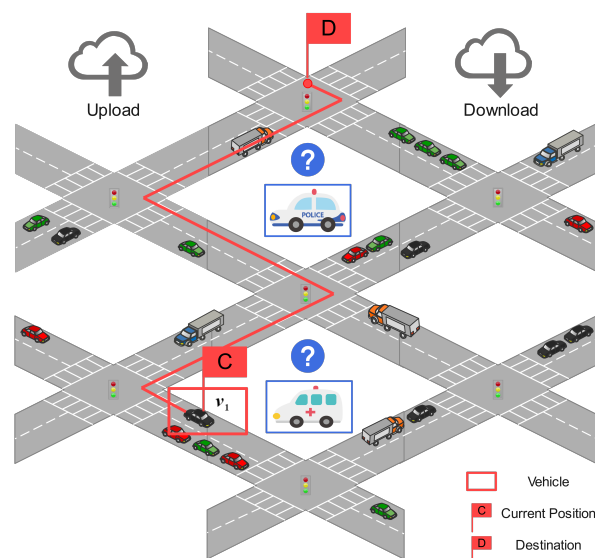


Figure 1. Service-oriented. Cooperative VRP in C-ITS.

Typically, the VRP is considered a variation of the traveling salesman problem, and the solutions for the VRP are almost entirely derived from solutions for the TSP [4–6]. Therefore, several classic static algorithms, covering the Dijkstra [7], Floyd [8], and Bellman–Ford [9] algorithms, have been widely adopted due to their good understandability. Although these methods are simple and effective, the shortcomings of these algorithms become quite obvious, including complex computing processes and local optimal results. To improve traveling efficiency, in recent decades, several heuristic and dynamic route planning mechanisms have been proposed, such as the A* [10], D* [11], rapidly exploring random trees [12], ant colony optimization [13], and evolutionary [14] algorithms. On this basis, Chabini et al. [10] proposed a method to find the fastest path with the A* algorithm, Stentz et al. [11] applied the D* algorithm to determine the lowest-cost path for robots, and Lavalle et al. [12] created an excellent and marvelous tree-searching algorithm for path finding. In addition, various AI methods have also been introduced to resolve this problem, including Neural Networks [15], fuzzy decision trees [16], reinforcement learning [17,18], and deep reinforcement learning [19,20]. Torki et al. [15] used a self-organizing Neural Network to solve the VRP, achieving outstanding performance. SongSang et al. [21] and Nazari et al. [17] designed successful VRP solutions through value-based and policy-based reinforcement learning accordingly. James et al. [19] explored how to combine online vehicle routing with offline deep reinforcement learning.

As aforementioned, vehicles with different service properties will always require different Qualities of Traveling (QoT), which is a key factor that must be considered in real transportation systems. Concretely, emergency vehicles, which have the highest QoT,

should encounter minimal travel delays to the greatest extent possible, while shuttle buses and private vehicles require medium QoT and the lowest QoT, respectively. Under such service-oriented situations, we introduced the term Service-oriented Cooperative ITS (SoC-ITS) [22], and the VRP becomes a Service-oriented Cooperative VRP (SoC-VRP). From the available literature, EVRP has attracted some attention in recent decades [23]. For planning an emergency path, Yang et al. [24] designed a situational grid road network model that maps spatiotemporal characteristics into dynamic road network graphs and uses regional features to represent road weights. Jotshi et al. [25] applied a related data fusion method to the EVRP under complex disaster circumstances. Özdamar et al. [26] designed a hierarchical clustering and routing procedure to address the EVRP during large disaster scenarios. Shelke et al. [27] developed a fuzzy prioritized control and management system to address the EVRP. Min et al. [28] determined a reliable estimated time of arrival and executed elastic signal preemption. Giri et al. [29] employed Dijkstra to calculate a combination of straight line and number of turns between the vehicle and the destination for emergency vehicle route planning. Jose and Grace [30] proposed a hybrid path optimization algorithm based on the fitness function which integrated the EWMA and the BSA algorithm. Li et al. [31] used road similarity and road label dependence as regularization items to optimize a path planning framework based on spatiotemporal data. Nguyen et al. [32] applied the method of evaluating the delay time of vehicles after clearing traffic obstacles on the path and issuing signals to preempt the timetable. Rout et al. [33] developed a congestion-aware IoT architecture based on open-source routers and fuzzy logic to guide emergency vehicles in smart city environments. Su et al. [34] presented a decentralized non-preemptive framework for simultaneous dynamic routing. Wen et al. [35] introduced the coevolutionary algorithm that uses the evolution mechanism to calculate the subpath weight function for emergency rescue path planning. Wu et al. [36] created an algorithm based on search and integer linear programming to clear a lane for nearby emergency lanes to ensure smooth and fast passage.

All visible studies have laid a good theoretical foundation in addressing such VRP and EVRP problems. However, these methods are all limited for Service-oriented and Coordinated VRP. On account of this, in this paper, a new route planning mechanism by employing a deep reinforcement learning mechanism is proposed. And the following contributions are mainly achieved. First, the new characteristics of SoC-VRP are analyzed and abstracted. And then, after comparing with different reinforcement learning methods, a value-based RL method is employed for solving such offline learning problems. Concretely, Rainbow DQN [37], a value-based single-agent deep reinforcement learning model issued by DeepMind, is applied to construct the solution method. On this basis, a hybrid prioritized route planning mechanism with specific state and reward functions is constructed. All these designs are implemented within the traffic simulator SUMO, and the performance is further verified.

The rest of this paper is organized as follows. In Section 2, several key issues are analyzed and abstracted; then, the solving principle with reinforcement learning is explained. In Section 3, a deep-reinforcement-learning-based mechanism for SoC-VRP is proposed and detailed. A series of experiments were conducted, and the performance of our work is discussed in Section 4. Section 5 presents the conclusions and future work.

2. SoC-VRP and Its Solving Principle with DRL

2.1. Analysis of SoC-VRP

Both VRP and SoC-VRP can also often be abstracted as special route searching problems over a Timed Directed Weighted Graph (T-DWG). In a T-DWG, a node v_j represents a road junction, and a weighted edge e_{jk} represents a direct road connection from v_j to v_k . The weight $w_{e_{jk}}$ of e_{jk} indicates the current situation of traffic flow, with a higher value indicating more traveling time on edge e_{jk} . Figure 2 shows such an example, which includes six nodes (i.e., junctions) {A, B, C, D, E, F}. When vehicle v_i is planning to travel from A to F, it will have six possible optional routes: <AC, CF>, <AC, CD, DF>, <AB, BD, DF>,

$\langle AB, BD, DC, CF \rangle$, $\langle AB, BE, ED, DF \rangle$, and $\langle AB, BE, ED, DC, CF \rangle$. Therefore, the optimal route planning for v_i should follow a continuous greedy decision-making process and be completed iteratively (i.e., $\langle AC, CD, DF \rangle$), where “greedy” means making each decision to select the road with the less traveling time during the full decision-making sequence when the time for traveling the rest of the road segments can be evaluated.

Based on this abstraction, the solving process of VRP can essentially be viewed as a progressive decision-making procedure to determine the sequence of road segments in a traffic network for vehicle v_i , in order to find a feasible route from its current position to the intended destination. As mentioned earlier, the complete decision procedure of SoC-VRP must adhere to a set of constraints, particularly considering the different QoT associated with vehicles of varying priorities. These constraints are in addition to the overarching considerations of travel safety and optimized efficiency for the entire traffic network. It is obvious, for SoC-ITS, that this problem will be more complex due to those particular constraints. Firstly, the solving process becomes significantly different from traditional route planning (Dijkstra, A^* , et al.), which is to plan the entire rest route greedily rather than decision-making greedily at only the current junction. The other important difference is that, during route planning, the route planning for vehicles with lower priorities should avoid to affect the routes both planned and being planned for vehicles with higher priority, guaranteeing the QoT of vehicles with higher priorities as much as possible. In this context, SoC-VRP also becomes a more complex cooperative planning problem.

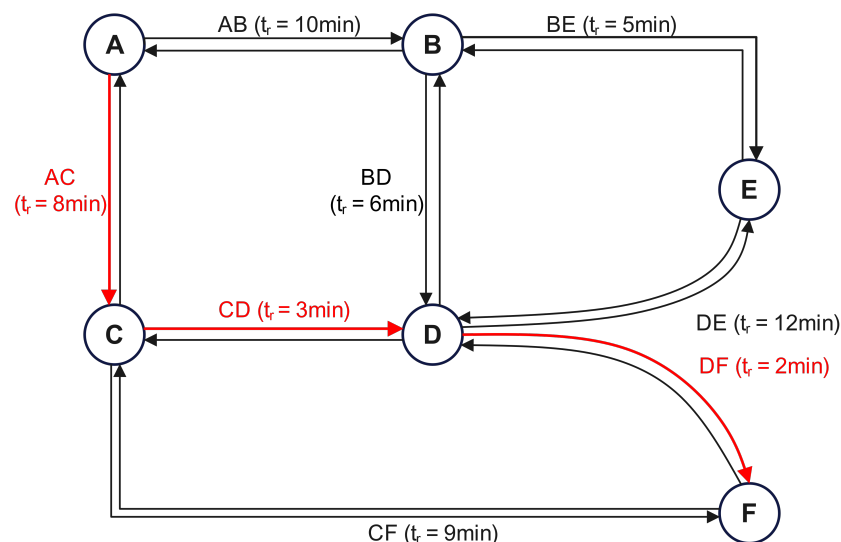


Figure 2. Mapping a Traffic Network to T-DWG.

Related to our work, in some studies based on DRL, urban (E)VRP is viewed as a search of the route that is only composed of junctions [19–21]. However, such methods still have difficulty solving the actual traffic problems, since they ignore the real-time traffic situation and the service attributes of vehicles. Indeed, our investigations have also provided empirical validation to support this assertion: the inclusion of junction as an integral component of the state yielded unsatisfactory results in terms of convergence within the reinforcement learning process. This deficiency stemmed from the propensity for nodes connected to the destination node to be erroneously perceived as antecedent nodes prior to arrival at the designated endpoint, consequently giving rise to potential disturbances. To rectify this limitation pertaining to road network representation, we proposed an advantageous approach involving the consideration of both termini of a given road segment as decisive factors for attaining the desired destination. As a result, we effectively mitigated this predicament by substituting junction-specific data with comprehensive road information, thereby augmenting the state representation and enhancing the learning process.

2.2. Reinforcement-Learning-Based Solving Model

Reinforcement Learning (RL) is widely recognized as a method rooted in the principles of the Markov Decision Process (MDP) [38,39], making it highly suitable for addressing sequential decision-making problems. The RL interactive framework consists of two essential components: the *Environment* and the *Agent*. The *Agent* determines an action based on the received states from the *Environment* and, subsequently, presents the decision-making action to the environment and receives feedback in the form of a reward. In the context of SoC-ITS, significant advancement in information capabilities, such as depth perception of the traffic *environment*, V2X-based data exchange, and global data fusion on traffic clouds, have enabled real-time determination of both the *Environment* and vehicle states. Consequently, a comprehensive observation of the overall traffic situation within a traffic-cloud-dominated zone becomes achievable. Notably, this property aligns inherently with the premise of an MDP, specifically emphasizing a fully observable environment. As a result, this rationale forms the basis for employing RL and its more powerful Deep Reinforcement Learning (DRL) counterparts in solving VRP and SoC-VRP. Leveraging RL and DRL models holds promise for enhancing the efficacy and effectiveness of these optimization tasks.

2.2.1. Fundamental Theory of DRL

An MDP model is specifically represented by a five-tuple $\langle S, A, R, P, \gamma \rangle$, as defined in Table 1. The collections of rewards and probabilities are denoted by R and P , respectively, with a state-space size of $S \times A \times S'$. Additionally, γ serves as an attenuated factor to refine the accumulated reward R_t^{Ac} , which represents the discounted total reward of the Markov chain after step t . Furthermore, the value function $Q(s, a)$ signifies the expected future accumulated reward for a specific state–action pair (s, a) within the MDP framework [38]. The reward and value functions can be formally defined as follows, where $\mathbb{E}$ represents expectation and R_t is reward function at step t :

Table 1. Parameter Definitions

Params	Definition
V	Vehicle agent
S/s	Space of states and State
A/a	Space of actions and Action
R_t	Reward function at step t
P	State transition probability matrix
γ	Attenuated factor
Q	Value function
T	Total number of steps
R_t^{Ac}	Accumulated discounted reward after step t
w/b	Weights and biases in the Neural Network
p	Position in the environment
pri	Priority of vehicles or roads
l_r	Length of road
n_r	Number of vehicles on the road
t_r	Expected time for vehicles passing this road
v	Velocity
ΔT	Time difference

$$\begin{aligned}
 Q(s, a) &= \mathbb{E} \left[R_t^{Ac} \mid S_t = s, A_t = a \right] \\
 &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}
 \end{aligned} \tag{1}$$

In conjunction with Equation (1), the Bellman expectation equation demonstrates the relationship between the value function of state s or the state–action pair (s, a) and the

value function of the subsequent state S_{t+1} within an MDP [40]. The derivation process is outlined as follows:

$$Q(s, a) = \mathbb{E}[R_{t+1} + \gamma(Q(S_{t+1}, A_{t+1})) \mid S_t = s, A_t = a] \quad (2)$$

Building upon the Bellman expectation equation, the MDP iteration method proves to be well-suited for decomposing complex problems into subproblems and iteratively determining suboptimal solutions through dynamic planning. Therefore, we adopt a value-based reinforcement learning model to address our specific problem. In this approach, the optimal target and optimal value function are updated as $v_{\pi^*} = \max(Q(s_i, a_i))$, where $s_i \in S$ and $a_i \in A$ [38]. This choice allows us to effectively optimize the learning process and derive desirable policies.

Deep reinforcement learning leverages the robust approximation capabilities of Deep Neural Networks (DNNs) to replace unwieldy Q-tables. In our study, this is achieved through Value Function Approximation with deep Q-networks (DQNs) [38,41,42], being a noteworthy contribution in this regard. It is important to highlight that value function approximation serves as an estimation method for large-scale reinforcement learning problems. By introducing additional parameters w and b in DNN, it becomes possible to determine the estimated value that aligns with the value function $Q(s, a)$. The formulation is as follows:

$$Q(s, a) \approx \hat{Q}(s, a, w, b) \quad (3)$$

Based on the analysis above, in this study, we employ a Deep Neural Network (DNN) as a function approximator, leveraging the differentiability of the state variable. Our main objective is to minimize the loss error between the actual value functions and their approximations. Building upon the principles of Temporal-Difference Learning (TDL), we utilize the TD target [38] to compute its error, which serves as a basis for gradient descent. The equation is presented as follows:

$$Q(s, a) = R + \gamma \hat{Q}(s', a', w, b) \quad (4)$$

2.2.2. Fundamental DRL Models for VRP and SoC-VRP

In this section, we present an integrated structure based on Deep Q-Networks (DQN) [42] that connects a reinforcement learning agent with a Deep Neural Network, as depicted in Figure 3. The agent primarily serves two fundamental functions. Firstly, it is responsible for selecting an action for the next step based on the observations of the environment and its current state. Initially, this function compares the value functions $Q(s, a)$ and selects the action with the highest value function or a random action following an ϵ -greedy policy. Subsequently, it interacts with the Neural Network to approximate the value function, enabling the acquisition of the value function of actions, i.e., $Q(s, a)$. The second function of the agent is to learn new knowledge from existing information and then utilize this knowledge to train itself.

Subsequently, we introduce batch learning, as illustrated in Equation (4), to refine the approximated value function, allowing us to estimate the value function of the subsequent state during the learning process. Once the TD error is obtained, the integrated agent employs this error to update the Neural Network's weights and biases through the gradient descent algorithm. Notably, the Deep Neural Network's functionalities are typically more immediate compared with those of the reinforcement learning agent, providing essential support to the latter. Specifically, the Deep Neural Network offers the integrated agent fundamental functions, determining outputs based on the inputs and fitting parameters according to the errors. Finally, we integrate the aforementioned elements and apply the Rainbow DQN [37] to our system to explore our scenarios (further details are provided in the following two paragraphs). This approach aims to enhance the performance of the deep reinforcement learning agent, encompassing the structuring of the overall framework

and analyzing additional improvements in the agents. These enhancements involve the target Neural Network, dueling network, and noisy parameters, as discussed in Section 3.

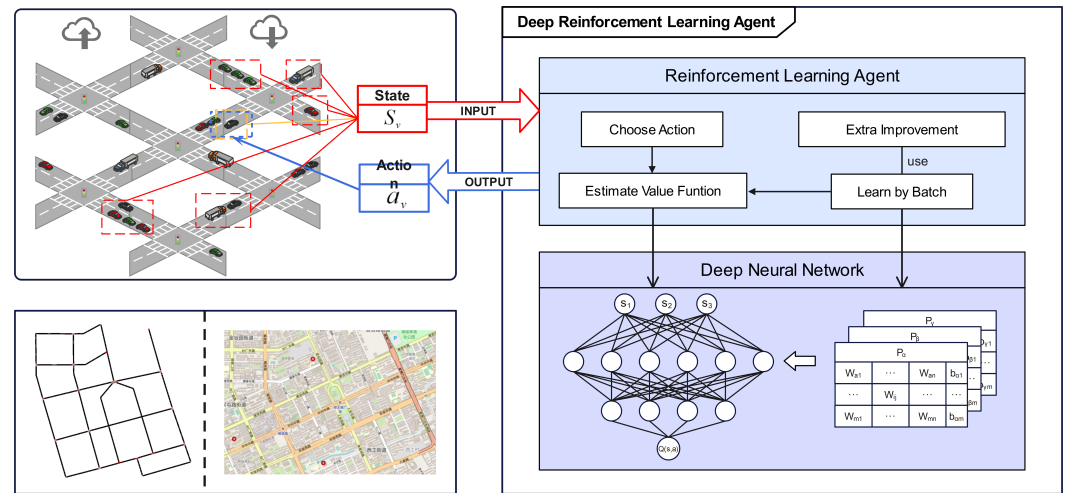


Figure 3. Deep reinforcement learning agent structure.

It should be clarified that the environment of ITS encompasses various traffic-related aspects, including traffic infrastructures and vehicles, as well as data and information. Based on the Markov Decision Process (MDP), the solution logic for solving the VRP can be represented as demonstrated in Figure 4, where the state space is defined. The previous route of a vehicle agent before reaching the current state S_D is marked with *, and the target state is S_F . The action space under S_D is denoted as $\{a_0, a_1, a_2, a_3\}$, and the corresponding rewards are $\{R_t^0, R_t^1, R_t^2, R_t^3\} = \{r_0, r_1, r_2, r_3\}$. Following the principle of value-based reinforcement learning, we assume that agents follow a greedy policy, typically an ϵ -greedy policy. The agent greedily chooses action a_3 , because its value function has the highest reward and best prediction in the Neural Network. Consequently, the agent transitions from state S_D to the next target state S_F . The state transition probability matrix P of S_D is depicted in Figure 4. Following this logic, the vehicle agent repeatedly makes action-choosing decisions until it reaches its destination.

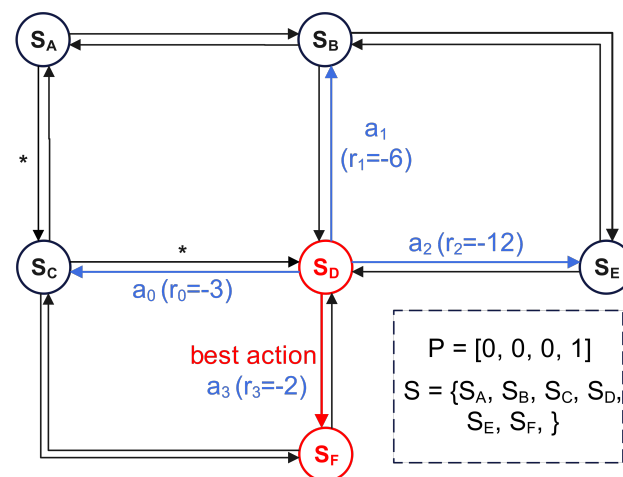


Figure 4. Reinforcement-learning-based routing principle.

For the SoC-VRP in SoC-ITS, we propose a specialized reinforcement learning agent with distinct state–reward s – r , as discussed in Section 3.2, to address the collaboration requirements and vehicle service properties. In C-ITS, route planning schemes must consider vehicle collaboration to a certain extent. As a result, we integrate global environment information, such as the number of vehicles on each road, into the state representation.

Additionally, valuable information, including service properties like vehicle priorities, is included in the state or reward to assist agents in maximizing the relative throughput of emergency vehicles in SoC-ITS.

3. A Novel DRL-Based SoC-VRP Solution Mechanism

In consideration of the deep reinforcement learning methodology, we devised a framework to address the SoC-VRP within the context of SoC-ITS. This framework, depicted in Figure 5, comprises three pivotal components: the *environment*, which serves as a prerequisite for the Markov Decision Process (MDP) model, established on a technical platform with the capability to interact with agents; the *deep reinforcement learning agent*, a central and critical component within the comprehensive mechanism responsible for action selection and learning, where each vehicle is treated as an individual agent; and the main control loop, serving as an intermediary between the *Environment* and the *Agent*, with its specific operating procedure outlined in the figure. The subsequent sections provide a comprehensive explanation of these components and their functionalities.

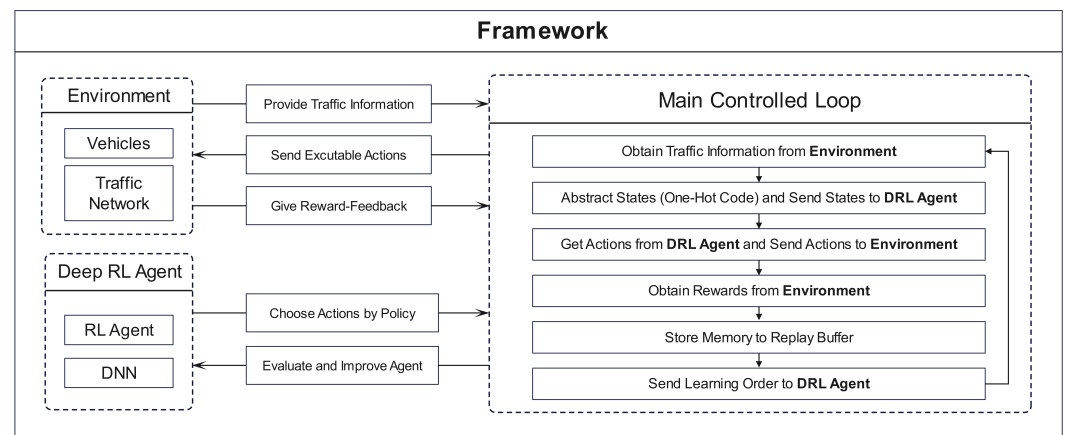


Figure 5. DRL Framework for Solving SoC-VRP.

3.1. Design of DRL “Environment”

In our proposed mechanism, the environment is designed to mirror an actual traffic scenario and carries out various responsibilities, including gathering state data, executing orders of actions, and providing immediate rewards to the agents. To fulfill its first responsibility, the environment acquires not only personal information of the agents but also real-time traffic information, akin to a data center using an RSU. This traffic information includes the number of cars on each road, behavior information of individual vehicles, and driving details of vehicles in each lane. Additionally, the personal information of each vehicle includes its coordinates in the traffic network, destination coordinates, identity, and service type. Furthermore, when a vehicle approaches a junction, it needs to make navigation decisions to choose the adjoining road. Hence, the environment must also relay the action orders given by the agents to the respective vehicle entities and ensure that the vehicles execute these actions. Ultimately, multiple immediate rewards are derived from the actions executed by the vehicle agents, and these immediate reward values are utilized by the agents to formulate a composite reward function.

Moreover, since we assume that all vehicles are connected smart vehicles and we are focused on macroroute planning, we simplify each road into a one-lane road to avoid interference from the lane-changing process and ensure a more intuitive route planning process. Considering such an environment as the scenario, we adopt an automatic dispatch mode for intersection management instead of using traffic signal lights. Additionally, we incorporate the concept of the decision zone, the purpose of which is to reduce computational burden while improving iteration efficiency [21]. In the presence of a significant number of vehicles operating in the traffic network, the cloud only allocates decisions for

vehicle agents within the decision zone, rather than those outside it. For a given road, the decision zone d_v can be represented as follows:

$$d_v = \min[l, v_{max} + v_{max}b(v)\tau_v] \quad (5)$$

where l and v_{max} denote the length of the road and the maximum speed of all vehicles, irrespective of their priority, respectively. The term b_v represents the deceleration function of the vehicles, and τ_v denotes the driver's reaction time, which is of lesser importance compared with the former parameters.

3.2. Design of DRL "Agent"

As a crucial component of the overall mechanism, deep learning and reinforcement learning are combined, enabling vehicle agents to acquire state and reward functions tailored to the specific SoC-ITS scenario. The DRL agent consists of two key components: the reinforcement learning component, which encompasses elements of the MDP model, primarily focusing on the state, action, and reward, and the deep learning component, which utilizes a Deep Neural Network model to enhance the performance of the aforementioned reinforcement learning component.

3.2.1. The "State" Model

In our proposed mechanism, the state represents an observation of the current environment from the perspective of each individual vehicle. Vehicles in our system have full access to environment observations. The state's parameters must accurately reflect the characteristics of the vehicle agents in various scenarios, enabling us to differentiate vehicle agents in different traffic situations by combining these parameters. For the SoC-VRP, a typical state for each vehicle agent is represented as the tuple $\langle p_v, p_t, n_r, t_r \rangle$, where p_v and p_t are the vehicle's current location and target location represented by coordinates, respectively. The other parameters have the same definitions as presented in Table 1. In contrast to SongSang's definition of t_r [21], we introduce the BPR (Bureau of Public Roads) road impedance function [43] to define t_r , which is given by

$$t_r = t_0 \times [1 + \alpha(\frac{Q_r}{C_r})^\beta] \quad (6)$$

Here, t_0 represents the time taken to traverse the road when it is clear, Q_r denotes the actual volume of vehicles on the road, C_r represents the capacity of the road, and α and β are impact factors. The recommended values by the US Highway Administration are $\alpha = 0.15$ and $\beta = 0.4$.

Furthermore, to accommodate different service properties of vehicle agents in SoC-ITS, we extend several parameters in the VRP solution, particularly for emergency vehicles. Based on our previous research [22,44], we define a priority value for each vehicle agent, which increases with higher priority levels. This means that high-priority vehicles have a much higher priority value than medium-priority vehicles, and medium-priority vehicles have a higher priority value than low-priority vehicles. The sum of the priority values of multiple vehicles is used to represent the weight of the road. Through the Grid Search method, we determine the specific priority values as follows: emergency vehicles have the highest priority value of 10, medium-priority vehicles have a priority value of 5, and other types of vehicles have a substantially lower priority value of 1, ensuring that lower-priority vehicles have minimal influence on the route planning of higher-priority vehicles. The state model can be expanded to the tuple $\langle p_v, p_t, n_r, t_r, pri_v, pri_r \rangle$ in the context of SoC-VRP, where pri_v represents the vehicle's priority value. Additionally, pri_r represents the sum of the priority values of all vehicles traveling on road r .

In Figure 6, the state $\langle p_v, p_t, n_r, t_r, pri_v, pri_r \rangle$ is observed by the ambulance agent within the solid-line rectangle marked by ev_1 . Here, p_v and p_t represent the coordinates of this vehicle and its target location (the point in the dotted-line rectangle marked by ev_1-t ,

(15, 30), and (45, 50)), respectively. The vehicle's priority is represented using one-hot encoding: [0, 0, 1] for low priority, [0, 1, 0] for medium priority, and [1, 0, 0] for high priority. The priority value for computing pri_r is assumed to be 15, because there is one ambulance and one bus on this road (** and * indicate a high-priority vehicle and a medium-priority vehicle separately, and no star indicates a low-priority vehicle). The number of vehicles on all roads, denoted by n_r , is represented by the vector [1, 2, 1, 2, 2, 4, 2, 0] (the elements are indexed counterclockwise from the left two roads in this example). The calculation of t_r is based on the BPR function and the environment settings and can be deduced from Equation (6) as follows:

$$\left. \begin{aligned} t_0 &= \frac{l_e}{v_{max}} \left\{ \begin{aligned} l_e &= 25 \\ v_{max} &= 13.89 \end{aligned} \right\} \\ Q_r &= [1, 2, 1, 2, 2, 4, 2, 0] \\ C_r &= 5 \end{aligned} \right\} \Rightarrow t_r \quad (7)$$

Finally, pri_r calculates the priority value for each road based on the vehicle types and numbers, resulting in a value of [1, 15, 1, 6, 6, 4, 11, 0], in accordance with the previously mentioned priority definition.

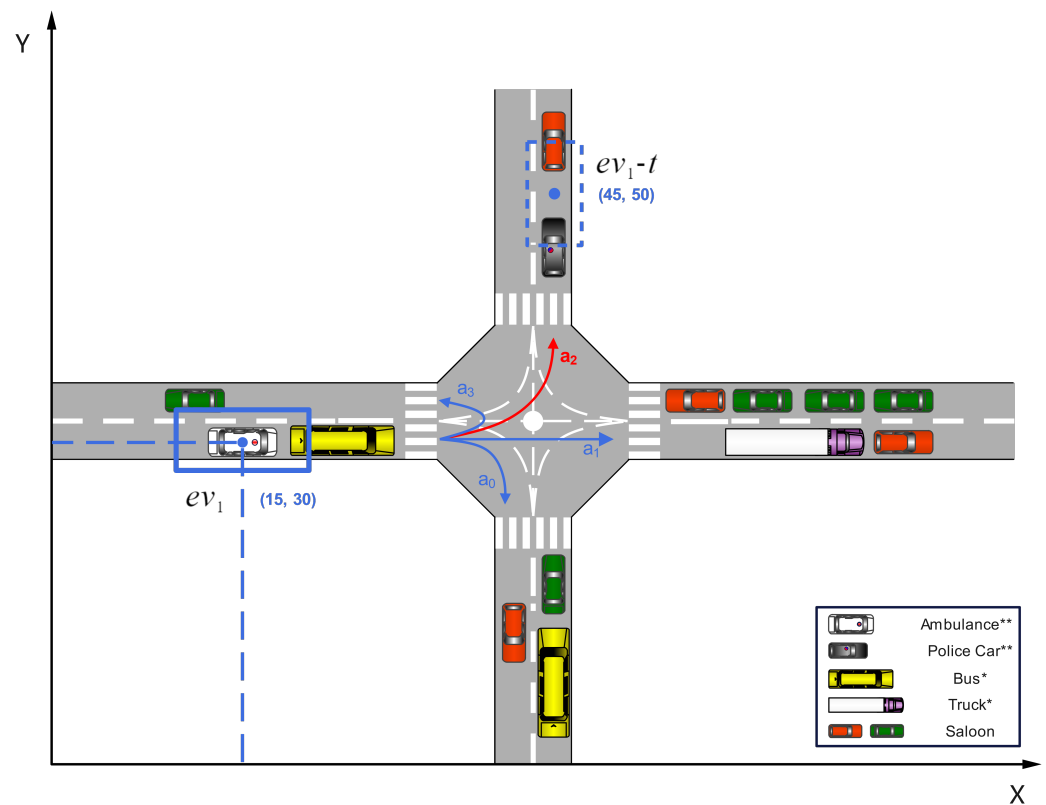


Figure 6. Various agents in SoC-ITS.

3.2.2. The “Action” Model

An action represents the turning decision made by a vehicle agent as it approaches an intersection. The agent selects an action from the executable action space of its current state while traveling. As illustrated in Figure 6, the ambulance agent has a 4-dimensional action space, denoted as $\{a_0, a_1, a_2, a_3\}$.

It is essential to emphasize that the action space used in VRP dynamically changes under different circumstances, a factor not mentioned in previous research. When approaching a crossroad, the agent has a 4-dimensional action space, while a 3-dimensional action space is applicable at a T-junction. To accommodate this dynamic-changing action

space requirement, we set the maximum number of connected roads for each road in the traffic network to match the size of the entire action space. For example, the agents in Figure 6 have a 4-dimensional action space when a crossroad is the largest intersection in the entire traffic network. However, when agents approach other types of intersections with a smaller number of connected roads than a crossroad, we utilize the action mask technique [45]. Unlike traditional methods, we multiply the predicted values of the Neural Network model (i.e., the Q values of the actions for a single state) by a tensor of the same length as the prediction tensor. In this tensor, certain values are set to 1, corresponding to the number of available connected feasible roads, and other optional values (usually $-\text{inf}$) represent the remaining number after deducting the number of 1s from the tensor's length. For instance, at a T-junction, the action mask tensor is $[1, -\text{inf}, 1, 1]$, where the indexing sequence (counterclockwise from the current road) represents turning right, traveling straight, turning left, and performing a U-turn. In this case, the value corresponding to traveling straight is set to $-\text{inf}$, because this action is not feasible.

3.2.3. Design of “Reward”

The reward is a crucial factor in reinforcement learning approaches, exerting a decisive effect on our system's performance. To enhance the learning efficiency of the agents, we designed the reward to be divided into two components: the normal reward r_n and the goal reward r_g .

The normal reward r_n is used to represent the reward for all decisions made by agents, excluding the final step when agents reach their target location. It is a complex and vital part of the total reward. As shown in Table 2, the composition of the normal reward varies in different traffic scenarios. Some elements in the normal reward used in ITS can also be considered in SoC-ITS, such as $D_{\|v \rightarrow \text{target}\|}$, which denotes the Euclidean distance between the vehicle and its destination, and $\Delta T(s_t, s_{t+1})$, representing the travel time of agents moving from state s_t to state s_{t+1} . Additionally, the reward elements designed for the SoC-ITS take into account a crucial characteristic: priority. Initially, we attempted to assign three independent and distinct normal rewards to vehicles with high, medium, and low priorities. This approach, however, was not effective in solving the SoC-VRP. Thus, we introduced the element Pri_{v_i} as the service-represented normal reward component. Pri_{v_i} represents the total priority of all vehicles behind vehicle v_i in the vehicle queue on the road, and its statistics are determined based on the vehicles on road V_r . This reward element evaluates whether taking a particular action causes congestion for other coordinated vehicles, particularly emergency vehicles, and its absolute value will increase when this action causes emergency vehicles to lag behind the executing vehicle. Conventionally, the normal reward r_n can be rescaled to a negative number for agents to learn with advantage. We designed various combinations of these elements, as shown in the table, and present the validation results in Section 4.

Table 2. Normal reward.

Scene	Reward Element	Value Illustration
ITS	$D_{\ v \rightarrow \text{target}\ }$	$\sqrt{400 + 900}$
	$\Delta T(s_t, s_{t+1})$	5
SoC-ITS	$\{Pri_{v_i} \mid v_i \in V_e\}$	1

The goal reward r_g is a positive reward given to agents to motivate them to reach their destination and should be defined based on the normal reward. According to Equation (1), we clip the value of normal rewards and observe that normal rewards in continuous steps

have a similar magnitude. Therefore, the normal rewards level off to a fixed numerical value r_n , allowing the accumulated reward R_t^{Ac} to be redefined as a geometric progression:

$$R_t^{Ac} = \sum_{k=0}^n \gamma^k R_{t+k+1} = \frac{1-\gamma^n}{1-\gamma} \times r_n \quad (8)$$

where n is the anticipated number of steps to reach the target. We then appropriately scale the goal reward r_g according to Equation (8). In cases where r_g is dissipated by an excessively large value of r_n (as r_g is usually positive), we utilize the following formulation:

$$r_g \geq \frac{1-\gamma^n}{1-\gamma} r_n \quad (9)$$

Finally, we determine the goal reward according to the minimum integer satisfying Equation (9).

Thus, the goal reward should be greater than 8.75.

Using the same example, consider the ambulance agent ev_1 within the solid-line rectangle in Figure 6; the normal reward for this agent is shown in Table 2. The distance between the vehicle agent and its destination $D_{\|v \rightarrow target\|}$ can be determined based on the coordinates of the vehicle itself, i.e., (15,30), and the coordinates of its destination, i.e., (45,20). The time to transfer from state s_t to state s_{t+1} is 5 seconds based on the simulation results. According to the previous definition of vehicle priority, Pri_{v_i} represents the sum of the priorities of all vehicles behind v_i belonging to vehicle collection V_r on road r , and the value of this parameter is equal to 0 because there is no vehicle behind v_i . If we use $\Delta T(s_t, s_{t+1})$ as our normal reward function and suppose that approximately $n = 3$ steps are needed to reach the target location, we can set γ to 0.5 (expectation on the future 3 steps). Thus, the goal reward should be greater than 8.75.

3.2.4. Neural Network Model

In this mechanism, we introduce the Neural Network (NN) model to address similar vehicle agents. The model takes the state as input and provides value functions for all actions in the action space to different vehicle agents as its output. Subsequently, each vehicle agent selects actions based on its respective value functions. The design of this NN model aligns with the principles of DRL, and as a result, this NN structure is deliberately kept simple and shallow. The rationale behind this choice lies in the fact that while a conventional deep learning model is utilized after the training process, the reinforcement learning model must learn and be employed alternately. Due to the high frequency of applying the model, excessively deep network structures may impede swift knowledge transfer.

According to Figure 5, this NN model can be evaluated and updated by initiating the main control loop. Our mechanism employs the traditional DQN method [42], specifically memory replay, to update the Neural Network. During the running interval, the deep RL agent intervenes in the replay process, allowing extraction of the MDP tuple elements and updates through loss functions such as mean square error (MSE) and the Huber function. When the distributional value function is not considered, we utilize the standard loss function, as depicted in Equation (10). On the other hand, when considering the distributional value function, we introduce the Kullback–Leibler (KL) divergence to calculate the error, resulting in the update of the loss function, as shown in Equation (11). The variables in Equations (10) and (11) are elaborated in Section 3.3.

$$\begin{aligned} L(\theta) &= \mathbb{E}_{\langle s,a,s' \rangle} [R_{\langle s,a \rangle} + \gamma Q_{\xi' \sim \xi'}(s', T(s'), \xi'; \theta) \\ &\quad - Q_{\xi \sim \xi}(s, a, \xi; \theta)]^2 \\ T(s') &= \arg \max_{a' \in A} Q_{\xi'' \sim \xi''}(s', a', \xi''; \theta^-) \end{aligned} \quad (10)$$

$$L(\theta) = \sum_{i=0}^{N-1} \frac{[\hat{z}_i]_{Q_{min}}^{Q_{max}}}{\Delta z} \log p_i(s, a) \quad (11)$$

$$\tau Z(s, a) = R_{\langle s, a \rangle} + \gamma PZ(s, a)$$

Additionally, we apply one-hot codes and normalize the mean and variance to improve the data processing technique of the NN model. We transform single discrete and finite values, such as pir_v , into their corresponding one-hot codes, thereby accelerating the learning of internal data features by the NN model. For example, the value 1 in 3 dimensions is converted to the one-hot code [1 0 0]. Normalization is designed to mitigate the difference between various orders of magnitude among different state elements. As the main control program runs, we synchronize the normalization of the mean and variance of the samples.

3.3. Improvement via Rainbow DQN

In this section, we present a brief overview of the advancements offered by Rainbow DQN [37], which are utilized to expedite convergence and optimize the results of our mechanism. SongSang [21] combined the widely used Double DQN [46] and Dueling DQN [47] models to address the VRP and simulate SUMO experiments. Although SongSang's approach demonstrates commendable performance, further enhancements to the method can be achieved.

Double DQN employs two NN models to handle the issue of overestimated value function $Q(s, a)$ in fully observed environments. An additional NN model, known as the target network, is used to select the action for the next state during experience replay, where the value function $Q(s, a)$ tends to increase infinitely. The dueling DQN model incorporates an advantage layer into the Neural Network, reducing differences in state-value functions among different actions and enabling more precise decomposition of the value function. The combination of these two variants constitutes the most significant modification to the DQN.

Another essential improvement is the implementation of Prioritized Experience Replay (PER) [48], which utilizes a nonstochastic sampling method. Newly added samples to the experience pool are assigned the highest TD error priority, allowing smaller TD error priority samples to be replayed more frequently, thereby preventing overfitting caused by value function approximation. Though this method deviates from the original empirical Independent and Identically Distributed (IID) nature of DQN, an annealing algorithm can be applied to mitigate this additional bias. Additionally, PER utilizes the sum tree data structure as an experience pool.

The multiple-step bootstrap [38] method has been proven to be effective in accelerating the convergence of reinforcement learning algorithms. We replace the reward parameter in the value function approximation with multistep rewards, similar to the n-step TD algorithm, as shown in Equation (12), where T represents the total number of steps until reaching the end, and k denotes the number of steps involved in the calculation. The value of k should be determined based on specific location circumstances, such as the traffic network.

$$G_t = R_{t+1} + \gamma Q(S_{t+1}, a_{t+1})$$

$$\Rightarrow \begin{cases} R_{t+1} + \gamma R_{t+2} + \dots + \gamma^k Q(S_{t+k}, a_{t+k}), & k < T \\ R_{t+1} + \gamma R_{t+2} + \dots + \gamma^{T-t-1} R_{end}, & k = T \end{cases} \quad (12)$$

The Noisy Network for Exploration [49], proposed by DeepMind, introduces additional parametric noise to the weights and biases of the Neural Network. This incorporation of noisy parameters successfully enhances network exploration by introducing uncertainty. Building upon this theory, we utilize factored Gaussian noise to introduce an individual layer before the output layer. The parameters of this layer, namely $\theta = (\mu + \sigma \odot \varepsilon)$, along with a pair of self-evolved μ and σ , contribute to exploring the optimal policy. In this theory,

μ is sampled from the distribution $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$, σ is set as $\frac{\sigma_0}{\sqrt{n}}$ (where σ_0 is suggested to be set to 0.5 by the author), n is the number of neurons, and arbitrary noise ξ is sampled from the scale ε when the model propagates forward or backward. The loss function of this enhancement is shown in Equation (10).

The Distributional Enhancement [50] is another contribution by DeepMind, which introduces the value function distribution atom Z into the reinforcement learning approximation, transforming the final outputs of the Neural Network model from fixed values to continuous distributions (bounded by predefined upper bound Q_{max} and lower bound Q_{min}). The fundamental component of this enhancement is the Bellman operator $\tau Z(s, a)$, and the loss in the model is evaluated using the KL divergence between categorical distributions, as suggested by the authors, instead of the Wasserstein metric. In practical applications, since the categories are defined before implementing the model, the target distribution entropy of each category is considered an invariant constant, allowing the use of the cross-entropy function to evaluate the difference among the distributions. The loss function of this enhancement is shown in Equation (11).

4. Experiments and Verification

In the absence of actual deployment of the C-ITS and SoC-ITS, we employed a scene-driven simulation method to validate the designs proposed in this work. The simulation was conducted using the extended simulator SUMO (Simulation of Urban Mobility) and a prototype of the traffic cloud. The selection of the SUMO simulator was based on its ability to accurately replicate authentic urban traffic conditions, with built-in vehicle entities that enable decision making by embedded vehicles.

4.1. Establishment of the Simulation Environment

SUMO provides the traffic control interface (Traci), which serves as an interface for running simulations and controlling entities such as vehicles and traffic lights. The software also offers user GUIs for visualization and verification, as demonstrated in Figure 7a,b, showing that the latter network has a larger size and road length compared with the former. To create a more realistic scenario, we introduced two real-world maps, shown in Figure 7c,e, obtained from OpenStreetMap [51]. We transformed these maps into two simulated traffic networks in SUMO, resulting in an enlarged version of the two traffic networks, as shown in Figure 7d,f. Detailed settings for these networks are provided in Table 3. These two traffic networks were utilized to assess the performance of different algorithms and mechanisms for addressing the VRP and SoC-VRP in terms of convergence speed and optimal results. To ensure realism, vehicle flows were randomly generated using the built-in function *randomTrip.py* in SUMO, which determined random vehicle positions and destinations. Moreover, in the SoC-VRP, to validate the experimental effects with greater realism, we assigned different hyperparameters to agents deployed on various traffic networks, as outlined in Section 3. The details of these hyperparameters are presented in Table 4. By employing these real-world maps and customizing hyperparameters, we aimed to create a more accurate and representative experimental environment for evaluating the performance of our proposed algorithms and mechanisms in addressing the SoC-VRP.

Table 3. Traffic network settings.

Setting	Network 1	Network 2	Network 3	Network 4
Edge quantity	14	22	36	54
Junction quantity	5	8	13	20
Average length	132.94	259.51	501.11	397.04
Allowed velocity	13	20	20	20

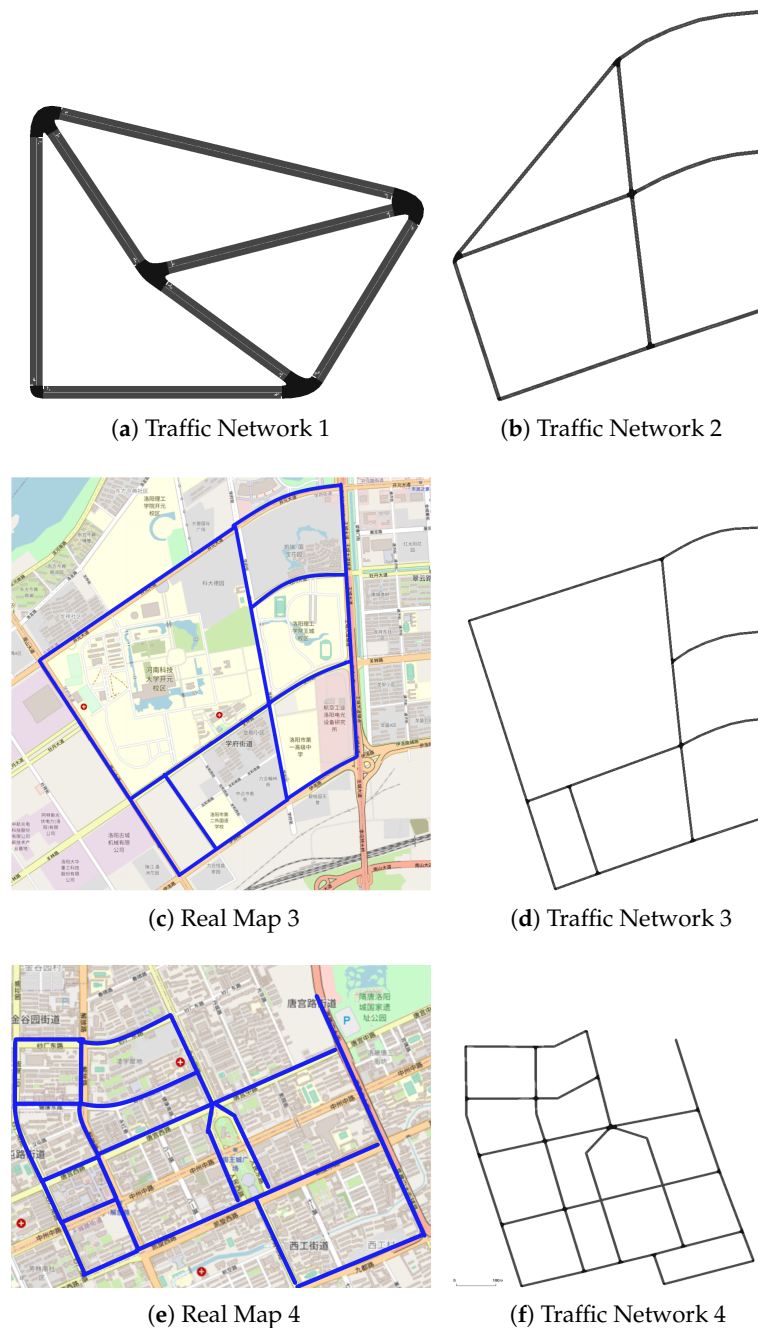


Figure 7. Maps and Networks in SUMO.

The evaluation metrics for this work mainly consist of the following aspects: Average Traveling Time (ATT), Average Waiting Time (AWT) at intersections, Average Traveling Velocity (ATV), and Time Loss (TL). **ATT** represents the time taken for a vehicle to reach its destination. **AWT** is the time that a vehicle spends waiting (when its velocity is less than 0.01 m/s) before making a turn. **ATV** is the average speed of the vehicle during its trip, calculated by dividing the total distance traveled by the total time taken. **TL** indicates the difference between the time it takes for the vehicle to travel at its maximum speed and the time it takes for the vehicle to travel at the actual simulated speed. It can be calculated as $t_{tot} \times (l_t - v_t)$, where t_{tot} is the total traveling time, and l_t and v_t denote the traveled distance and velocity at each time step, respectively. These evaluation metrics serve as essential indicators to assess the performance and efficiency of our proposed mechanisms in addressing traffic flow optimization and VRP. By considering these metrics, we can gain

valuable insights into the effectiveness and impact of the implemented strategies on overall traffic management and congestion reduction.

Table 4. Hyperparameter settings.

Hyperparameters	Network 1	Network 2	Network 3	Network 4
T-network update/learning	500	800	1000	1000
Gamma	0.5	0.6	0.7	0.8
Multisteps	2	3	5	8
Learning rate	0.0001			
Batch size	128			
Memory size	100,000			
Number of atoms	51			
PER attenuating α	0.6			
PER calculating β	0.4→1			
Noisy initial σ_0	0.5			

Additionally, we included A* [10] and SongSang’s methodology [21] (which follows the same principle as the dueling double DQN framework) as our comparative route planning baseline algorithms to evaluate their performance for the VRP. For the A* algorithm, the heuristic searching metric was set to the travel time of the vehicles.

4.2. Experiments and Analysis

4.2.1. Comparison of Variants Addressing the VRP

We conducted an evaluation to assess the effects of different improvements applied in the DQN model on addressing the VRP in Traffic Network 1 (Figure 7a). To ensure better control of variables, we imported only one vehicle priority type (private car) into our system. The evaluation results are depicted in Figure 8, where the red, green, blue, yellow, pink, and orange lines represent the Dueling-DDQN model, Rainbow models without a distributional value function, without multibootstrapping, without a noisy network, without prioritized experience replay, and a hybrid Rainbow model, respectively. The results of this evaluation allow us to compare and analyze the performance of various DQN model improvements in addressing the VRP in Traffic Network 1.

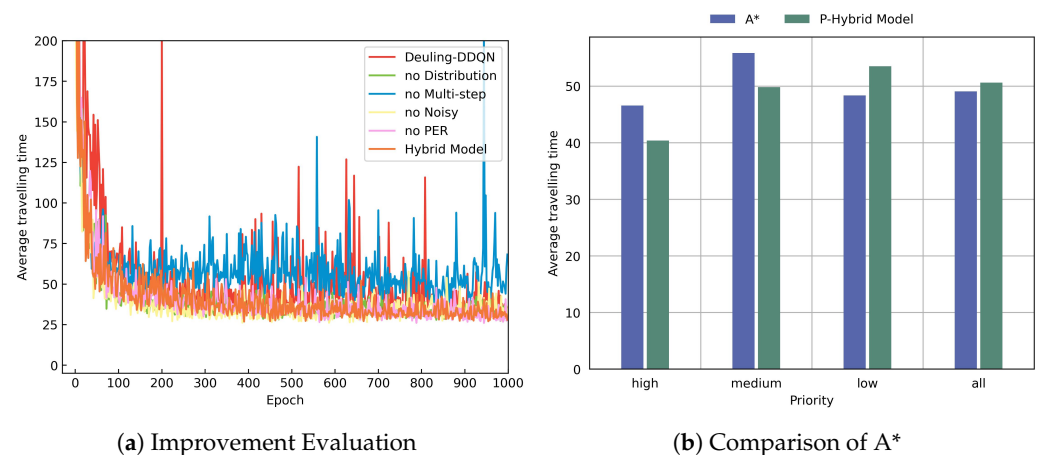


Figure 8. VRP effects.

The trends of the curves in the results were verified by comparing them with those of the original Rainbow DQN model [37]. Our hybrid Rainbow model demonstrates the best performance and stability when applied to the VRP. Firstly, the figure illustrates that the variants perform similarly in terms of convergence speed compared with the Rainbow DQN model. The Dueling-DDQN, models without PER, and models without noisy features show the slowest convergence speeds. On the other hand, the other variants and our hybrid Rainbow model converge at similar speeds. Secondly, in terms of optimal degree performance, our experimental results slightly differ from those of the Rainbow DQN model. The Dueling-DDQN and models without multistep converge to a local optimal result. In contrast, the final ATTs of the other models are closer and smaller than those of the aforementioned two models, particularly the hybrid model. Unlike Rainbow DQN, the performance of the model without PER does not show a similar trend to the model without multistep in our test. The original experience replay converges quickly through stochastic sampling, because the experimental exploration space, with a finite vehicle number and typical size traffic network, is not large enough to reveal the superiority of PER. Thirdly, we assessed stability to evaluate model performance. Our hybrid model exhibits the highest stability, and the models without PER and without noise show similar results, because these two improvements enhance efficiency rather than the overall effect. Appropriate parameters can be set for the models without distributions and without multisteps to ensure that these models exhibit excellent performance for exploring effects, similar to our model without distributions.

However, the model without multisteps exhibits a discrepancy from the Rainbow DQN result. Analyses suggest that this phenomenon is caused by the multistep parameter (2 in Table 4), which is set too large, leading agents to consider excessive steps before decision making. For instance, a vehicle could reach its destination either through the fastest one-step process or through multistep two-step decisions, which may cause disturbances in the decision-making process.

4.2.2. Optimized Effects in the SoC-VRP

On the same maps, with both a single random flow and ten random flows, we conducted an evaluation to assess the performance of our proposed prioritized mechanism in addressing SoC-VRP.

- (i) Single random flow;
- (ii) Ten random flows.

In the **single** random flow experiment, we tested A* and our proposed prioritized mechanism on Network 1 (Figure 7a) for the SoC-VRP. The evaluation was verified by the ATT, illustrated in Figure 8. For this experiment, the number of vehicles was set to 50, with a flow distribution consisting of 5% high-priority vehicles, 15% medium-priority vehicles, and 80% low-priority vehicles, making up the total 50 vehicles. Notably, our model involved randomly training the Neural Network for 1000 epochs, repeated ten times, and we took the mean value of the training results to achieve more reliable and convincing outcomes. The results demonstrate that our proposed mechanism for routing high-priority and medium-priority vehicles proved to be considerably more efficient than the A* algorithm. However, it was observed that our mechanism caused the ATT of low-priority vehicles to increase. This finding indicates that our prioritized mechanism successfully prioritizes the routing of high- and medium-priority vehicles, but this prioritization can have an impact on the travel time for low-priority vehicles. Further analysis and fine-tuning of the prioritization strategy may be necessary to balance the efficiency of high- and medium-priority vehicles while minimizing the negative effects on low-priority vehicle travel time. Such optimization efforts could lead to more equitable and efficient routing solutions across all vehicle priorities in the SoC-VRP.

Next, we introduce Traffic Network 1 and perform the same training processes to evaluate the experimental criteria on **ten** different vehicle flows with the same flow distribution as the single random flow. By computing the mean index values of the three kinds

of vehicles, we can evaluate whether our proposed prioritized mechanism is superior to other algorithms, such as A*, dueling double DQN [21], and the Rainbow DQN hybrid model [37], in addressing the SoC-VRP. The results of this experiment are shown in Figure 9, and to solve the SoC-VRP on Traffic Network 1 using a Neural Network, each algorithm is trained and converges with an upper bound of 3000 epochs. The Dueling-DDQN model does not demonstrate any advantage on the SoC-VRP, as it shows higher ATT, AWT, and TL values for high-priority vehicles than for low-priority vehicles. Additionally, this model exhibits higher ATTs for all vehicle types compared with the results obtained from A*. It possesses the worst optimal performance and the slowest convergence speed in 3000 epochs among all the comparison methods. On the other hand, the other two mechanisms (the hybrid model and our proposed mechanism) display lower ATT, AWT, and TL values and higher ATV for high- and medium-priority vehicles, indicating better performance than the other models. Our proposed mechanism emerges as the best-performing one. Furthermore, the reduction in ATT, AWT, and TL and the increase in ATV for medium-priority vehicles are notably lower compared with high-priority vehicles. In contrast, low-priority vehicles show a different trend, and both the hybrid Rainbow model and our proposed mechanism also improve the ATT values for private cars more than the A* algorithm. Additionally, our proposed mechanism outperforms the hybrid model in this regard.

However, when the prioritized mechanism is applied to low-priority vehicles, it marginally increases the ATV while decreasing the AWT and TL, which differs from the result of the hybrid model. This discrepancy arises because the proposed mechanism optimizes the global traffic situation by reducing congestion, as depicted in the fourth column that considers all vehicles in Figure 9.

In conclusion, in both single-flow and multiple-flow scenarios, due to Rainbow DQN's outstanding performance in convergence, our proposed mechanism significantly reduces the ATT for high-priority vehicles and slightly reduces the ATT for medium-priority vehicles in the traffic network, thereby feasibly solving the SoC-VRP.

4.2.3. Larger Traffic Network Verification

We conducted experiments on Traffic Networks 2–4 (Figure 7b,d,f) to assess the performance of our mechanism compared with traditional algorithms in addressing the SoC-VRP for larger and more unpredictable traffic networks. These experiments involved ten randomly distributed flows for Network 2 and five randomly distributed flows for Networks 3 and 4. Traffic Network 2 consisted of 50 vehicles, with the same vehicle priority proportions as mentioned before: 5% high-priority, 15% medium-priority, and 80% low-priority vehicles. On the other hand, Traffic Networks 3 and 4 had 100 vehicles, with proportions of high-, medium-, and low-priority vehicles set at 5%, 10%, and 85%, respectively. The results for Networks 2–4 are depicted in Figure 10, with the maximum number of epochs set to 5000. Further details are provided in Figures A1–A3. It is worth noting that the Dueling-DDQN algorithm did not perform well and failed to converge in Network 4, resulting in the absence of data for this algorithm in the third row of Figure 10.

Analyzing the overall trends in Figure 10, it becomes evident that as the traffic network becomes larger and contains more vehicles, and the Dueling-DDQN algorithm becomes an unfeasible solution. In contrast, our proposed hybrid model and prioritized mechanism continue to function, although their performance is not as impressive as in the Network 1 results, especially for the hybrid model. It is easy to find that the hybrid model performs worse from Networks 1 to 4; the difference value between it and the A* algorithm becomes smaller, and even its ATT mean value surpasses A*'s in Network 4. The hybrid model's mean values of ATT, AWT, and TL for all vehicles increase when facing a larger network and higher vehicle flow. On the other hand, our proposed prioritized mechanism still performs well for high-priority vehicles, but the same improvement trend is not evident for medium- and low-priority vehicles in the figures. The improvement in passing quality for high-priority vehicles comes at the cost of sacrifices from medium- and low-priority

vehicles, leading to higher ATT, AWT, and TL values and lower ATV values for these vehicle categories.

Comparing the results of Network 2 with Network 1 and the results of Network 4 with Network 3, we observe that as the traffic network size increases, our hybrid model and mechanism exhibit weaker performance. Additionally, the quantity of vehicles serves as another reason for this decrease in performance.

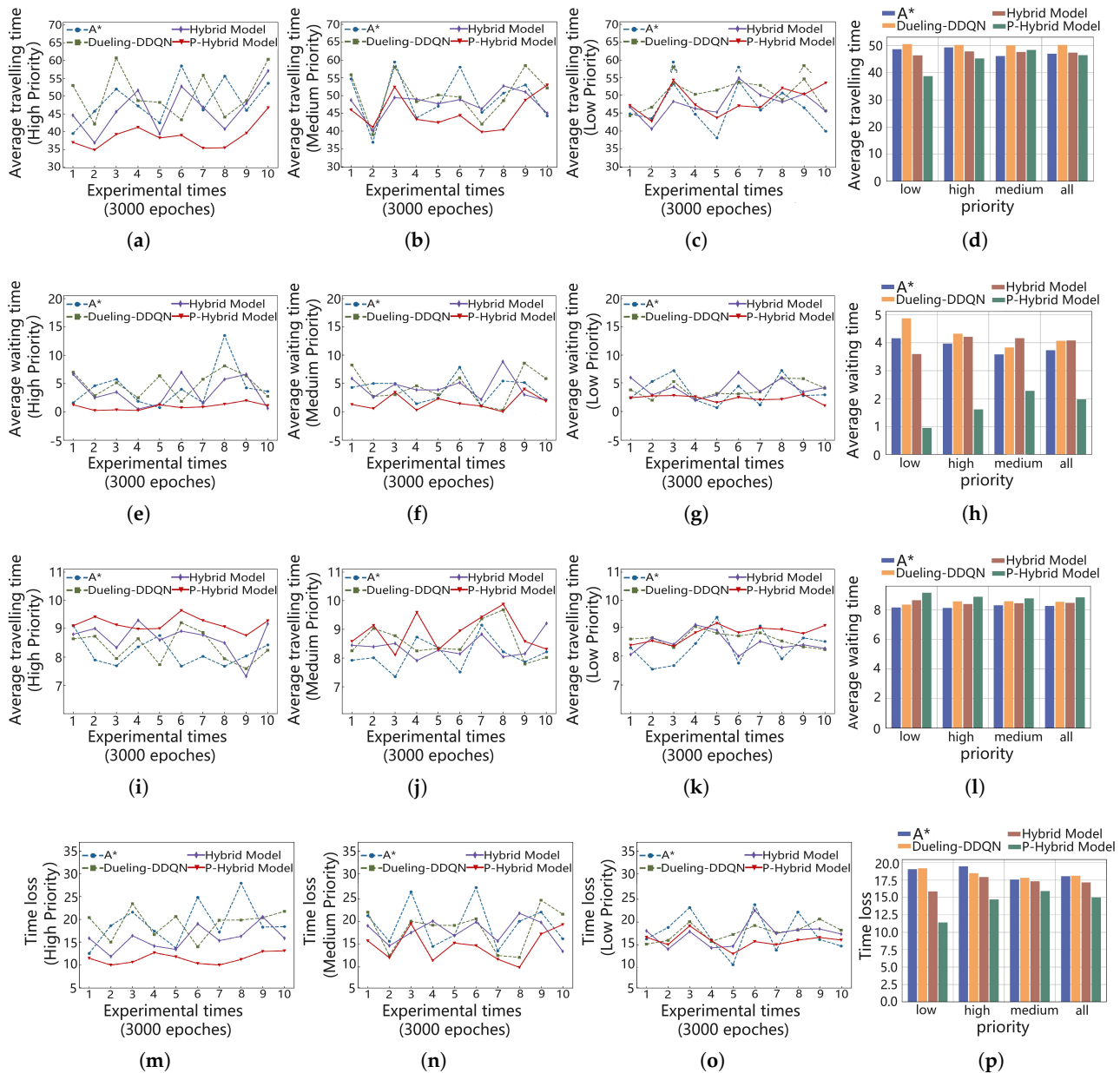


Figure 9. SoC-VRP results on Road Network 1: (a) high-priority ATT; (b) medium-priority ATT; (c) low-priority ATT; (d) ATT algorithm comparison; (e) high-priority AWT; (f) medium-priority AWT; (g) low-priority AWT; (h) AWT algorithm comparison; (i) high-priority ATV; (j) medium-priority ATV; (k) low-priority ATV; (l) ATV algorithm comparison; (m) high-priority TL; (n) medium-priority TL; (o) low-priority TL; (p) TL algorithm comparison.

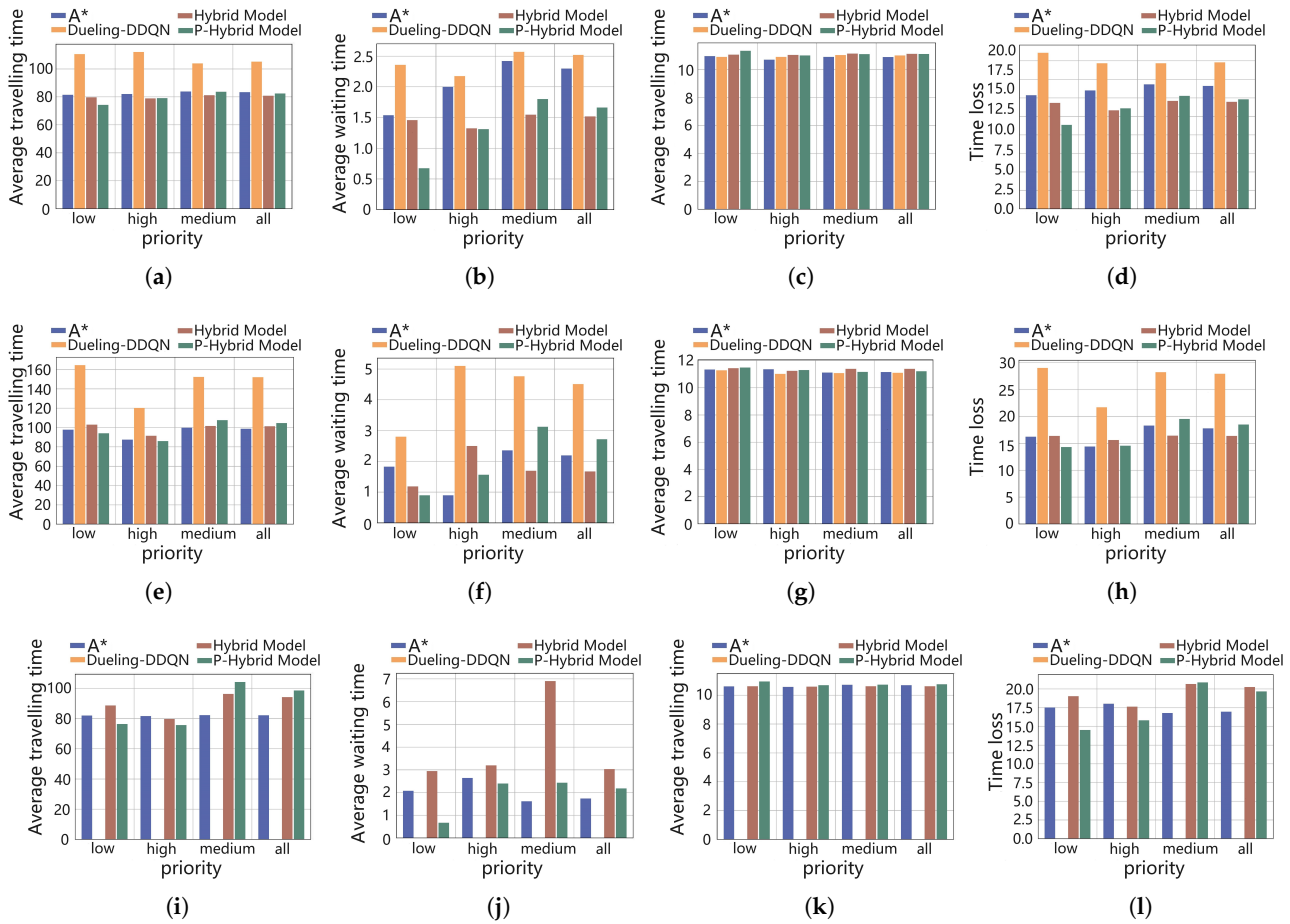


Figure 10. SoC-VRP results on larger road networks: (a) ATT comparison in Network 2; (b) AWT comparison in Network 2; (c) ATV comparison in Network 2; (d) TL comparison in Network 2; (e) ATT comparison in Network 3; (f) AWT comparison in Network 3; (g) ATV comparison in Network 3; (h) TL comparison in Network 3; (i) ATT comparison in Network 4; (j) AWT comparison in Network 4; (k) ATV comparison in Network 4; (l) TL comparison in Network 4.

Despite the larger and more complex traffic networks, our proposed mechanism maintains good performance for emergency vehicles due to our particular state–reward design concerning priority for the Markov decision process chain. The proposed method effectively decreases crucial ATT and TL values and increases ATV values more than the A* algorithm. However, as the traffic network becomes larger and more complex, the optimized effect of our proposed prioritized mechanism for SoC-VRP gradually diminishes, and the training time increases when solving the SoC-VRP in larger traffic networks. Further optimizations and adjustments may be required to achieve better performance and efficiency for the prioritized mechanism in such scenarios.

4.2.4. Vehicle Proportion and Quantity Comparison

Furthermore, we perform two other experiments to assess the performance of our mechanism in Network 1 (Figure 7a) and Network 2 (Figure 7b).

- Increasing vehicle proportion experiments;
- Increasing number of vehicle experiments.

The first set of experiments is presented in Figure 11, illustrating the changes in ATT as the proportion between high-priority and non-high-priority vehicles varies. Networks 1 and 2, with 50 and 200 vehicles, respectively, were used, considering the different sizes of the traffic networks. The deployment performance on different types of vehicles changes as

the proportion of high-priority and non-high-priority vehicles changes. We observe that the effectiveness of our proposed mechanism for solving the SoC-VRP declines as the number of non-high-priority vehicles decreases. The threshold ratios of 5:15 and 6:14 were observed for Network 1 and 2, respectively. Different traffic networks hold different approximate proportion thresholds. Before reaching the ratios of 5:15 and 6:14, our mechanism performs well, but its effectiveness decreases as the proportion increases. When the proportion of emergency vehicles increases, the number of other types of vehicles decreases, resulting in a decrease in the difference between different-priority vehicles.

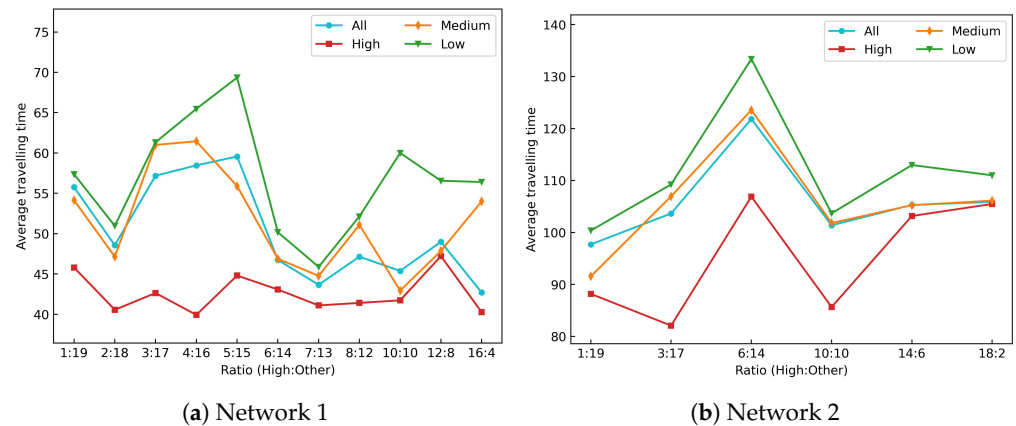


Figure 11. Experiment Results on Different Proportions of Vehicle Service Properties.

The second set of experiments aimed to evaluate the influence of the number of vehicles on model convergence in Figure 12. We varied the number of vehicles for Networks 1 as 30, 50, and 80 and for Network 2 as 50, 100, 150, 200, and 250. The results indicate that the convergence speed and the final convergence effect of our proposed mechanism decrease, and convergence becomes more challenging as the number of vehicles increases.

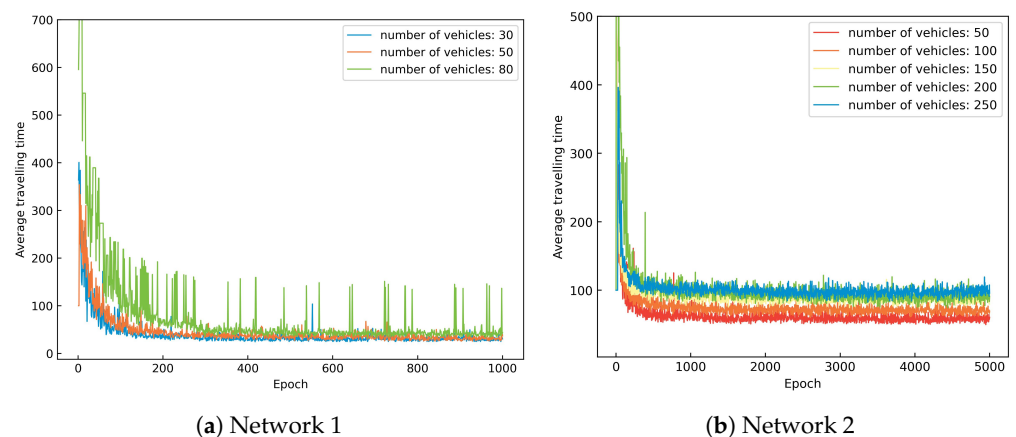


Figure 12. Convergence situation of established DRL.

5. Conclusions

The optimization of VRP is a challenging task, particularly in emergency situations where delays and congestion can have significant impacts, especially for emergency vehicles. In this paper, we proposed a mechanism that combines the VRP with deep reinforcement learning to address these challenges. The characteristics of this work are that this proposed mechanism not only reduces redundant computations in traditional algorithms but also integrates vehicle priority to effectively solve the SoC-VRP. The experimental results show it is effective. On the one hand, we improve the Dueling-DDQN method to

Rainbow DQN for solving the VRP, improving the convergence speed and results. On the other hand, introducing the vehicle priority allows us to successfully plan different types of routes for various types of vehicles, especially for emergency vehicles, and our proposed mechanism shows better results than the other three methods. In addition to the above, when the MDP combined with models in automotive-grade [52], this work is also applied for the economical, energy-saving operation of road vehicles and also affects the reduction in environmental emissions.

In future work, several improvements can be made to our proposed mechanism. Firstly, the mechanism should be retrained when transferred to different traffic networks due to the current deficiency of Deep Reinforcement Learning (DRL) in handling various environments. Creating a ubiquitous environment that can abstract characteristics and data from any type of traffic network could address this limitation and enhance the stability of reinforcement learning. Secondly, adapting our mechanism to real-world scenarios with signal-controlled traffic would require a long training process and realistic traffic data. Overcoming this challenge would be crucial for practical deployment. Lastly, exploring multiagent deep reinforcement learning solutions may offer promising ways to address asynchronous decision making among multiple agents in multivehicle route planning scenarios.

Author Contributions: Conceptualization, B.H. and K.Z.; methodology, B.H. and K.Z.; software, B.H. and Q.L.; validation, B.H., Z.G. and J.Z. (Junle Zhou); formal analysis, B.H. and K.Z.; investigation, B.H., J.Z. (Jiahao Zhang) and J.Z. (Junle Zhou); resources, B.H. and K.Z.; data curation, B.H. and J.Z. (Junle Zhou); writing—original draft preparation, B.H.; writing—review and editing, K.Z. and A.d.L.F.; visualization, B.H. and Q.L.; supervision, K.Z.; project administration, K.Z.; funding acquisition, K.Z. and A.d.L.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (61972318, 61572403), the Fundamental Research Funds for the Central Universities (3102019ghxm019), Shaanxi Provincial Science and Technology Project (2023-GHZD-47).

Data Availability Statement: The data that support the findings of this study are available on request from the corresponding author, KaiLong Zhang. The data are not publicly available due to them containing information that could compromise research participant privacy.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

SoC-VRP Service-oriented Cooperative Vehicle Routing Problem

Appendix A

The appendix contains detailed figures of supplementary data for experiments in Section 4.2.3; in Section 4.2.3, we only show the summary charts. The data of Traffic Networks 2–4 are shown as follows Figures A1–A3.

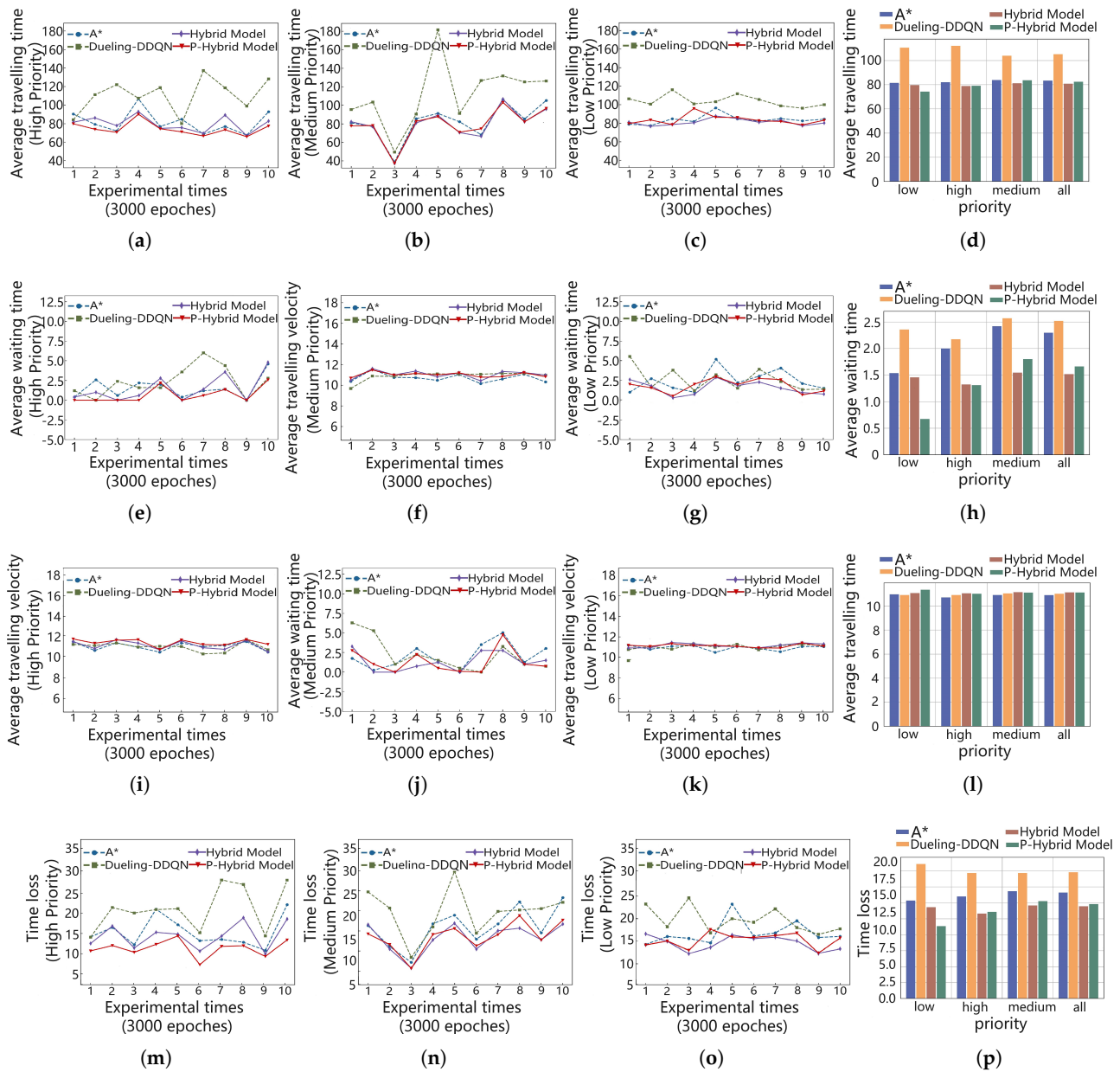


Figure A1. SoC-VRP results on Network 2: (a) high-priority ATT; (b) medium-priority ATT; (c) low-priority ATT; (d) ATT algorithm comparison; (e) high-priority AWT; (f) medium-priority AWT; (g) low-priority AWT; (h) AWT algorithm comparison; (i) high-priority ATV; (j) medium-priority ATV; (k) low-priority ATV; (l) ATV algorithm comparison; (m) high-priority TL; (n) medium-priority TL; (o) low-priority TL; (p) TL algorithm comparison.

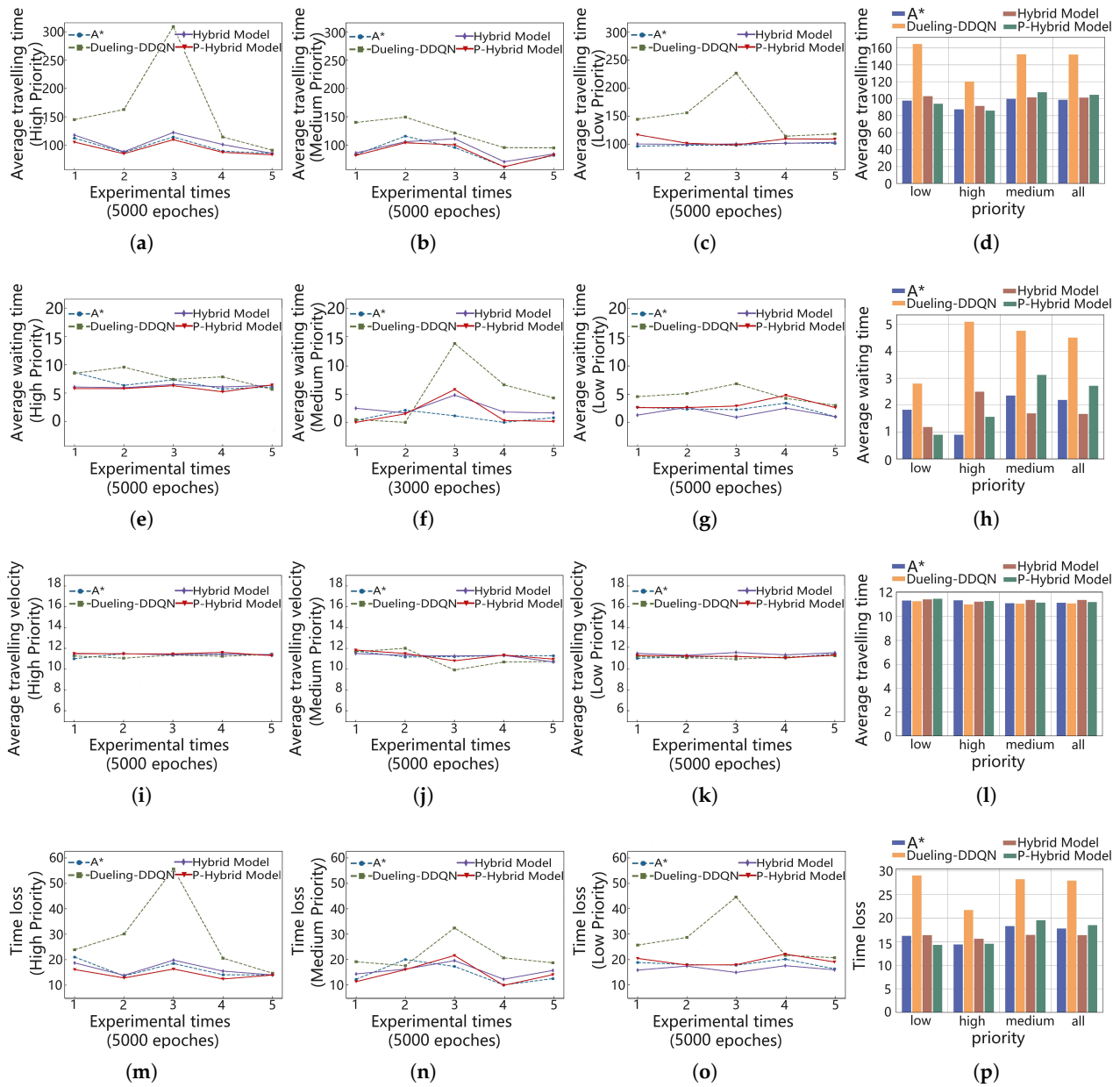


Figure A2. SoC-VRP results on Network 3: (a) high-priority ATT; (b) medium-priority ATT; (c) low-priority ATT; (d) ATT algorithm comparison; (e) high-priority AWT; (f) medium-priority AWT; (g) low-priority AWT; (h) AWT algorithm comparison; (i) high-priority ATV; (j) medium-priority ATV; (k) low-priority ATV; (l) ATV algorithm comparison; (m) high-priority TL; (n) medium-priority TL; (o) low-priority TL; (p) TL algorithm comparison

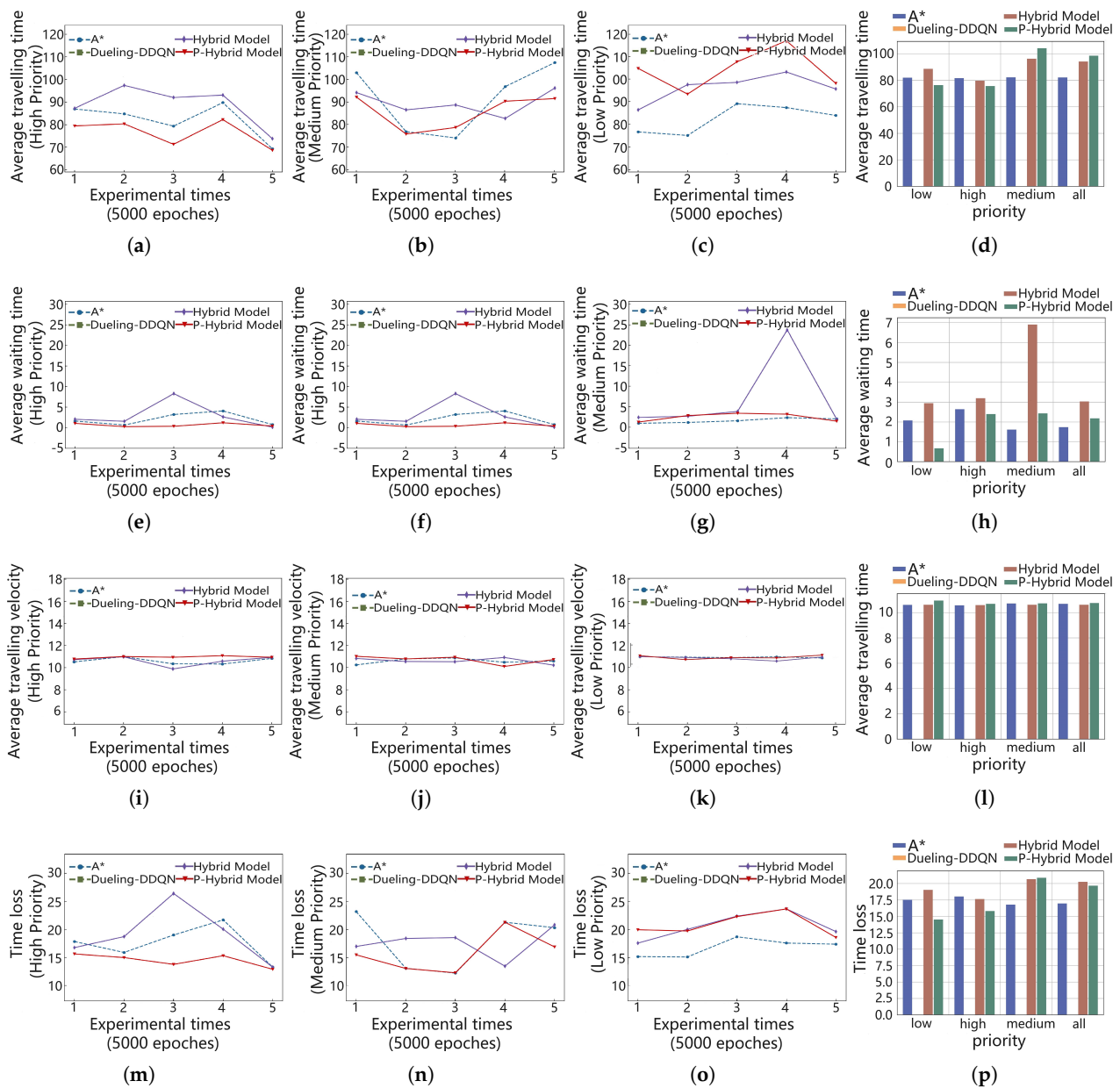


Figure A3. SoC-VRP results Network 4: (a) high-priority ATT; (b) medium-priority ATT; (c) low-priority ATT; (d) ATT algorithm comparison; (e) high-priority AWT; (f) medium-priority AWT; (g) low-priority AWT; (h) AWT algorithm comparison; (i) high-priority ATV; (j) medium-priority ATV; (k) low-priority ATV; (l) ATV algorithm comparison; (m) high-priority TL; (n) medium-priority TL; (o) low-priority TL; (p) TL algorithm comparison.

References

1. Laporte, G. The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Oper. Res.* **1992**, *59*, 345–358. [\[CrossRef\]](#)
2. Toth, P.; Vigo, D. *The Vehicle Routing Problem*; SIAM: Philadelphia, PA, USA, 2002.
3. Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a large-scale traveling-salesman problem. *J. Oper. Res. Soc. Am.* **1954**, *2*, 393–410. [\[CrossRef\]](#)
4. Gambardella, L.M.; Dorigo, M. Ant-Q: A Reinforcement Learning approach to the traveling salesman problem. In *Machine Learning Proceedings 1995*; Frieditis, A., Russell, S., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 1995; pp. 252–260. [\[CrossRef\]](#)
5. Bello, I.; Pham, H.; Le, Q.V.; Norouzi, M.; Bengio, S. Neural Combinatorial Optimization with Reinforcement Learning. *arXiv* **2016**, arXiv:1611.09940.

6. Liu, F.; Zeng, G. Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Syst. Appl.* **2009**, *36*, 6995–7001. [CrossRef]
7. Imran, A.; Salhi, S.; Wassan, N.A. A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *Eur. J. Oper. Res.* **2009**, *197*, 509–518.
8. Wang, J.; Sun, Y.; Liu, Z.; Yang, P.; Lin, T. Route planning based on floyd algorithm for intelligence transportation system. In Proceedings of the 2007 IEEE International Conference on Integration Technology, Shenzhen, China, 20–24 March 2007; pp. 544–546.
9. Eisner, J.; Funke, S.; Storandt, S. Optimal route planning for electric vehicles in large networks. In Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 7–11 August 2011.
10. Chabini, I.; Lan, S. Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks. *IEEE Trans. Intell. Transp. Syst.* **2002**, *3*, 60–74.
11. Stentz, A. The focussed d* algorithm for real-time replanning. In Proceedings of the IJCAI, Montreal, QC, Canada, 20–25 August 1995; Volume 95, pp. 1652–1659.
12. LaValle, S.M. Rapidly-Exploring Random Trees: A New Tool for Path Planning. 1998. Available online: <https://api.semanticscholar.org/CorpusID:14744621> (accessed on 5 September 2023).
13. Bell, J.E.; McMullen, P.R. Ant colony optimization techniques for the vehicle routing problem. *Adv. Eng. Inform.* **2004**, *18*, 41–48.
14. Bederina, H.; Hifi, M. A hybrid multi-objective evolutionary optimization approach for the robust vehicle routing problem. *Appl. Soft Comput.* **2018**, *71*, 980–993.
15. Torki, A.; Somhon, S.; Enkawa, T. A competitive neural network algorithm for solving vehicle routing problem. *Comput. Ind. Eng.* **1997**, *33*, 473–476.
16. Du, J.; Li, X.; Yu, L.; Dan, R.; Zhou, J. Multi-depot vehicle routing problem for hazardous materials transportation: A fuzzy bilevel programming. *Inf. Sci.* **2017**, *399*, 201–218.
17. Nazari, M.; Oroojlooy, A.; Snyder, L.; Takác, M. Reinforcement learning for solving the vehicle routing problem. *arXiv* **2018**, arXiv:1802.04240.
18. Lu, H.; Zhang, X.; Yang, S. A learning-based iterative method for solving vehicle routing problems. In Proceedings of the International conference on learning representations, New Orleans, LA, USA, 6–9 May 2019.
19. James, J.; Yu, W.; Gu, J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3806–3817.
20. Zhao, J.; Mao, M.; Zhao, X.; Zou, J. A hybrid of deep reinforcement learning and local search for the vehicle routing problems. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 7208–7218.
21. Koh, S.; Zhou, B.; Fang, H.; Yang, P.; Yang, Z.; Yang, Q.; Guan, L.; Ji, Z. Real-time deep reinforcement learning based vehicle navigation. *Appl. Soft Comput.* **2020**, *96*, 106694. [CrossRef]
22. Zhang, K.; Yang, A.; Su, H.; de La Fortelle, A.; Miao, K.; Yao, Y. Service-Oriented Cooperation Models and Mechanisms for Heterogeneous Driverless Vehicles at Continuous Static Critical Sections. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1867–1881. [CrossRef]
23. Zhang, X.; Yu, X.; Wu, X. Exponential Rank Differential Evolution Algorithm for Disaster Emergency Vehicle Path Planning. *IEEE Access* **2021**, *9*, 10880–10892. [CrossRef]
24. Yang, B.; Yan, J.; Cai, Z.; Ding, Z.; Li, D.; Cao, Y.; Guo, L. A novel heuristic emergency path planning method based on vector grid map. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 370.
25. Jotshi, A.; Gong, Q.; Batta, R. Dispatching and routing of emergency vehicles in disaster mitigation using data fusion. *Socio-Econ. Plan. Sci.* **2009**, *43*, 1–24.
26. Özdamar, L.; Demir, O. A hierarchical clustering and routing procedure for large scale disaster relief logistics planning. *Transp. Res. Part E Logist. Transp. Rev.* **2012**, *48*, 591–602.
27. Shelke, M.; Malhotra, A.; Mahalle, P.N. Fuzzy priority based intelligent traffic congestion control and emergency vehicle management using congestion-aware routing algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *2019*, 1–18.
28. Min, W.; Yu, L.; Chen, P.; Zhang, M.; Liu, Y.; Wang, J. On-demand greenwave for emergency vehicles in a time-varying road network with uncertainties. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3056–3068.
29. Giri, A.R.; Chen, T.; Rajendran, V.P.; Khamis, A. A Metaheuristic Approach to Emergency Vehicle Dispatch and Routing. In Proceedings of the 2022 IEEE International Conference on Smart Mobility (SM), New Alamein, Egypt, 6–7 March 2022; pp. 27–31. [CrossRef]
30. Jose, C.; Vijula Grace, K. Optimization based routing model for the dynamic path planning of emergency vehicles. *Evol. Intell.* **2022**, *15*, 1425–1439. [CrossRef]
31. Li, X.; Niu, X.; Liu, G. Spatiotemporal representation learning for rescue route selection: An optimized regularization based method. *Electron. Commer. Res. Appl.* **2021**, *48*, 101065. [CrossRef]
32. Nguyen, V.L.; Hwang, R.H.; Lin, P.C. Controllable Path Planning and Traffic Scheduling for Emergency Services in the Internet of Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 12399–12413. [CrossRef]
33. Rout, R.R.; Vemireddy, S.; Raul, S.K.; Somayajulu, D.V. Fuzzy logic-based emergency vehicle routing: An IoT system development for smart city applications. *Comput. Electr. Eng.* **2020**, *88*, 106839. [CrossRef]

34. Su, H.; Zhong, Y.D.; Dey, B.; Chakraborty, A. Emvlight: A decentralized reinforcement learning framework for efficient passage of emergency vehicles. In Proceedings of the AAAI Conference on Artificial Intelligence, Washington DC, USA, 7–14 February 2022; Volume 36, pp. 4593–4601.
35. Wen, H.; Lin, Y.; Wu, J. Co-Evolutionary Optimization Algorithm Based on the Future Traffic Environment for Emergency Rescue Path Planning. *IEEE Access* **2020**, *8*, 148125–148135. [[CrossRef](#)]
36. Wu, J.; Kulcsár, B.; Ahn, S.; Qu, X. Emergency vehicle lane pre-clearing: From microscopic cooperation to routing decision making. *Transp. Res. Part B Methodol.* **2020**, *141*, 223–239.
37. Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; pp. 3215–3222.
38. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
39. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
40. Bellman, R. On the theory of dynamic programming. *Proc. Natl. Acad. Sci. USA* **1952**, *38*, 716.
41. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
42. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [[CrossRef](#)]
43. US Bureau of Public Roads; Office of Planning; Urban Planning Division. *Traffic Assignment Manual for Application with a Large, High Speed Computer*; US Department of Commerce: Washington, DC, USA, 1964.
44. Zhang, K.; Zhang, D.; de La Fortelle, A.; Wu, X.; Gregoire, J. State-driven priority scheduling mechanisms for driverless vehicles approaching intersections. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2487–2500. [[CrossRef](#)]
45. Huang, S.; Ontañón, S. A Closer Look at Invalid Action Masking in Policy Gradient Algorithms. *arXiv* **2020**, arXiv:2006.14171.
46. van Hasselt, H.; Guez, A.; Silver, D. Deep Reinforcement Learning with Double Q-Learning. *Proc. AAAI Conf. Artif. Intell.* **2016**, *30*. [[CrossRef](#)]
47. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling Network Architectures for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York City, NY, USA, 19–24 June 2016; Balcan, M.F., Weinberger, K.Q., Eds.; PMLR: New York, NY, USA, 2016; Volume 48, pp. 1995–2003.
48. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
49. Fortunato, M.; Azar, M.G.; Piot, B.; Menick, J.; Osband, I.; Graves, A.; Mnih, V.; Munos, R.; Hassabis, D.; Pietquin, O.; et al. Noisy Networks for Exploration. *arXiv* **2017**, arXiv:1706.10295.
50. Bellemare, M.G.; Dabney, W.; Munos, R. A Distributional Perspective on Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; Precup, D., Teh, Y.W., Eds.; PMLR: New York, NY, USA, 2017; Volume 70, pp. 449–458.
51. Haklay, M.; Weber, P. Openstreetmap: User-generated street maps. *IEEE Pervasive Comput.* **2008**, *7*, 12–18. [[CrossRef](#)]
52. István, L. An integrated analysis of processes concerning traffic and vehicle dynamics, development of laboratory applying real traffic scenarios. In Proceedings of the 2016 ASME/IEEE International Conference on Mechatronic and Embedded Systems and Applications (MESA), Auckland, New Zealand, 29–31 August 2016.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.