telecom

MDPI

Article

Deep Reinforcement Learning for Multi-Objective Optimization of Latency-Constrained V2X Communications

Sadia Islam [1], Saniul Alam [1], Muhammad Khandaker [2], Risala Tasin Khan [1], ASM Ashraf Mahmud [3,*] and Faisal Tariq [4]

[1] Institute of Information Technology, Jahangirnagar University, Savar 1342, Bangladesh; risala@juniv.edu (R.T.K.)
[2] Main Roads Western Australia, East Perth, WA 6892, Australia; ruhul.khandaker@mainroads.wa.gov.au
[3] School of Computing, Engineering and Physical Sciences, University of the West of Scotland, Paisley PA1 2BE, UK
[4] James Watt School of Engineering, University of Glasgow, Glasgow G12 8QQ, UK; faisal.tariq@glasgow.ac.uk
* Correspondence: ashraf.mahmud@uws.ac.uk

Abstract

Meeting the stringent latency requirements of safety-critical vehicular services is a major challenge for beyond fifth generation (B5G) systems, and rapidly varying channel state information (CSI) in device-to-device (D2D) enabled vehicle-to-everything (V2X) communication makes the problem harder. This paper addresses power allocation in a D2D-enabled vehicular network in which vehicle-to-vehicle (V2V) links reuse the spectrum of vehicle-to-infrastructure (V2I) links. The problem is formulated as a non-convex optimization problem that maximizes the V2I sum capacity subject to a V2V latency constraint. We use deep reinforcement learning to jointly select the V2I and V2V transmit powers from large-scale channel information, trained from measured outcomes rather than from the closed-form channel expressions used by the optimization-based solution, and a normalized preference weight that trades V2I capacity against V2V latency within a single network. The framework is implemented with a discrete-action deep Q-network (DQN) and with a continuous-action deep deterministic policy gradient (DDPG) agent. Under the nominal channel model, the DQN attains 85–95% and the DDPG agent 90–99% of the sum capacity of the model-based optimum, with a per-block computation time two to three orders of magnitude lower. When the reported large-scale channel gains contain errors of 2 dB, the model-based solution violates the V2V latency constraint on 46–56% of the links, whereas agents trained under such errors keep the violation rate below 0.5%. DDPG is preferable when the channel model is accurate, whereas the DQN trains more reliably under CSI errors. Queue-level simulations further show that, at the considered traffic loads, queueing rather than transmission failures dominates the delay tail.

Keywords: V2X communications; power allocation; deep reinforcement learning; deep Q-network; DDPG; multi-objective reinforcement learning; D2D; latency

Check for updates

Academic Editor: Seyeong Choi

Received: 8 August 2026
Revised: 2 November 2026
Accepted: 6 November 2026
Published: 9 November 2026

Copyright: © 2026 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.

1. Introduction

In vehicular communication systems, vehicles act as communicating nodes that can exchange information both with neighboring vehicles and with the base station (BS). Tasks that require large amounts of data and frequent Internet access are best supported by high-capacity vehicle-to-infrastructure (V2I) links, whereas safety-critical information is best carried by vehicle-to-vehicle (V2V) links. In this paper, we consider a cellular vehicular network that supports both V2V and V2I links, referred to as a vehicle-to-everything

(V2X) network, which is expected to be the backbone of emerging intelligent transport infrastructure [1,2]. Each vehicle can connect to the BS through a V2I link, while V2V links connect neighboring vehicles, as shown in Figure 1.

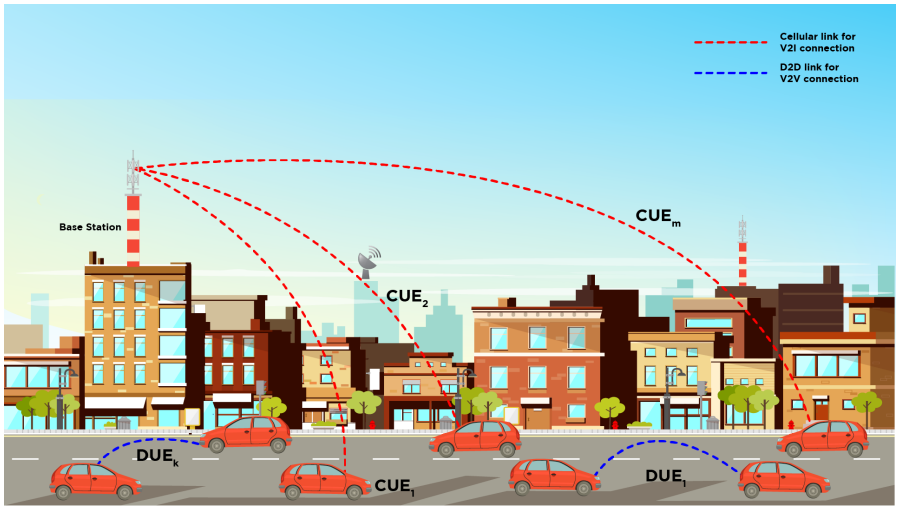


Figure 1. D2D-enabled vehicular communications for both V2I and V2V links. Each cellular user equipment (CUE) holds a V2I link; each D2D user equipment (DUE) pair forms a V2V link that may reuse the resource block of one CUE.

To realize the full potential of autonomous vehicles, vehicular communication must provide very low latency and high reliability, because a delay of milliseconds in receiving information can lead to life-threatening accidents. Ultra-reliable and low-latency communication (URLLC) is therefore considered a crucial part of vehicular communication in 5G and emerging 6G systems [3]. V2V communication mainly carries driving-safety information exchanged with neighboring vehicles, which requires very low latency (typically around 1 ms) as well as high reliability. High node mobility causes rapidly changing channel conditions, which makes it difficult to maintain both latency and reliability targets. Finding an efficient resource-allocation strategy that meets stringent latency requirements therefore remains a major challenge for V2X deployment [4,5].

The work in [6] proposed a separate resource block and power allocation algorithm for V2X networks. The authors translated the latency and reliability requirements of V2X communication into optimization constraints and maximized the CUE sum rate while satisfying the reliability and latency constraints of the DUEs; to cope with rapidly changing CSI, only slowly varying CSI is used. Similarly, ref. [7] formulated the latency and reliability requirements as constraints that depend only on slowly varying fading information and, since the problem is NP-hard, proposed a radio resource management scheme that maximizes the CUE sum rate under the safety constraints of the DUEs. In [8], a two-timescale resource-allocation strategy based on Lyapunov stochastic optimization was used for power and resource block allocation in V2V communication, reducing queueing latency and improving reliability. The authors of [9] jointly optimized frame size and resource allocation in V2V URLLC systems. The work in [10] shared the spectrum of one V2I link among several V2V links using graph-based matching. In [11], the problem of delayed CSI feedback was considered. The authors of [12] computed the power allocation of every possible CUE–DUE pair under a DUE reliability constraint and used the Hungarian method to select the pairs. The works [13,14] extended [12] to add a latency constraint by modelling the queueing and retransmission of packets at the DUE and proposed an iterative algorithm that minimizes latency while fulfilling the QoS requirements; latency-violation-probability and effective-capacity formulations followed [15,16]. Because URLLC packets

are short, Shannon-rate models underestimate the error probability, and finite-blocklength analysis [17,18] has recently been used for V2V resource allocation [19]. In 5G New Radio (NR) V2X, BS-controlled resource allocation corresponds to sidelink Mode 1, whereas Mode 2 relies on autonomous resource selection [4,20].

Although much work has been done on resource allocation in V2X networks, conventional optimization methods have difficulty fully addressing the requirements. These methods are often formulated as NP-hard problems for which only locally optimal solutions can be found, and the resulting iterative algorithms can be computationally demanding [21,22]. Another limitation of existing optimization approaches is that they require exact models, which abstract away many imperfections of reality for mathematical convenience; if the model is inaccurate, the solution may be inaccurate as well. In addition, these solutions are computed at a central controller, which must solve a per-pair optimization problem for every candidate CUE–DUE pair whenever the reported large-scale information changes. This online computation grows with the number of vehicles. The scheme studied in this paper is also centralized at the BS and uses the same large-scale CSI reports as the optimization-based solution of [13], so it does not reduce the signaling overhead. What it changes is the online computation, which becomes a forward pass of a trained network, and the reliance on an exact channel model.

Reinforcement learning (RL) has shown great potential for decision making in emerging wireless systems [23], even when an exact mathematical model is not available [24]. In RL, the reward can be designed to reflect the objective, and the agent learns a strategy by interacting with the environment. The authors of [25] combined fountain codes with machine-learning-based channel allocation to meet URLLC requirements. The work in [26] used supervised learning to approximate the WMMSE algorithm with much lower complexity. The work in [27] used a multi-agent DQN in which each V2V link is an agent that selects its spectrum and power; the network determines the power of the DUEs only and assumes that CUEs always transmit at maximum power to keep the action space small. The work in [28] also used multi-agent RL with an additional latency constraint and likewise assumed fixed maximum CUE power. In [29], platoon leaders act as agents that learn their power allocation with a double DQN. Since transmit powers are continuous, ref. [30] used an actor-critic approach, and ref. [31] used a deep deterministic policy gradient (DDPG) framework for reliable power and resource block allocation. The work in [32] used two-timescale DRL for mode selection and resource allocation, and ref. [33] used RL for user scheduling and power control in NOMA-based V2X. More recently, meta-learning has been combined with DQN and DDPG for fast adaptation in dynamic V2X environments [34,35], multi-agent actor-critic methods have been applied to NR-V2X resource allocation [20], and classical and learning-based V2X resource allocation have been compared systematically [5]. Event-triggered DRL has been used to allocate power and blocklength in URLLC V2X links with fewer learning updates [36], explainable-AI tools have been used to reduce the state inputs of multi-agent DQN-based V2X allocation under finite-blocklength reliability constraints [37], and graph neural networks have been combined with double DQN for distributed V2V resource allocation [38]. A recent benchmark of multi-agent RL for V2X resource allocation reports that actor-critic methods outperform value-based methods on the most demanding tasks [39], which is consistent with our comparison in Section 4. Table 1 summarizes the related works, which also demonstrates the novelty of the proposed work.

In this paper, we consider the existing problem formulation of [13] and use deep reinforcement learning (DRL) as an alternative to its iterative per-pair power allocation, keeping the Hungarian method for spectrum pairing. The power allocation of both V2V and V2I links is considered, whereas [27,28] assume that V2I links transmit at maximum power. We use a single-objective agent that maximizes the system capacity subject to the

V2V latency constraint, and a multi-objective agent that additionally reduces the V2V latency. The framework is implemented with two learners: a discrete-action DQN and a continuous-action DDPG agent, which share the state, reward, and training procedure. The main differences from existing DRL-based V2X power allocation are: (i) joint V2I–V2V power control, which we show is necessary when the V2V constraint is tight; (ii) training from measured transmission outcomes, so that the learned policy does not depend on the closed-form channel model used by the optimization-based solution; (iii) a normalized multi-objective reward whose preference weight is an input of the network, so that one network covers the whole capacity–latency trade-off; and (iv) an evaluation of the delay distribution, not only of its mean. The novelty lies in this formulation and its evaluation rather than in a new neural network architecture or learning algorithm.

Table 1. Summary of closely related resource-allocation methods for D2D-enabled V2X networks.

Work	Method	Powers Optimized	V2V QoS Metric	CSI Used	Learning-Based
[12]	Closed form + Hungarian	V2I and V2V	Outage probability	Large-scale	No
[13,14]	Iterative search + Hungarian	V2I and V2V	Mean sojourn time	Large-scale	No
[15]	Model-based optimization	V2I and V2V	Latency-violation probability	Large-scale	No
[8]	Lyapunov, two-timescale	V2V	Queue-length tail	Mixed	No
[19]	Block coordinate descent	V2V (power, blocklength)	Finite-blocklength error	Large-scale	No
[28]	Multi-agent DQN	V2V only (V2I fixed)	Latency deadline	Local observations	Yes
[27]	Multi-agent DQN	V2V only (V2I fixed)	Payload delivery probability	Local observations	Yes
[29]	Multi-agent double DQN	V2V (platoons)	Delivery probability	Local observations	Yes
[34]	DQN + DDPG, meta-RL	V2V	Rate/delivery	Local observations	Yes
This work	Single-/multi-objective DQN and DDPG + Hungarian	V2I and V2V (joint)	Mean sojourn time; delay tail evaluated	Large-scale (possibly erroneous)	Yes

The main contributions of this paper are summarized as follows:

- A DRL-based solution is developed for the joint V2I–V2V power allocation problem of a D2D-enabled V2X network under a V2V latency constraint, with a well-defined decision process and training procedure.
- Both single-objective and multi-objective agents are designed; the multi-objective reward is normalized, and its preference weight is studied (sojourn time in ms and failure probability).
- Simulation results compare the DQN and DDPG implementations with the model-based optimum of [13] and a DQN with fixed CUE power, and report the delay distribution, the improvement in scalability and computation time and, most importantly, the robustness to errors in the large-scale CSI. They also compare the two learners and identify when each is preferable.

The rest of the paper is organized as follows. Section 2 presents the system model and problem formulation. Section 3 gives an overview of DRL and describes the proposed DQN- and DDPG-based resource allocation. Section 4 presents and analyses the simulation results, and Section 5 concludes the paper.

2. System Model

We consider a D2D-enabled vehicular network with M pieces of cellular user equipment (CUEs) requiring V2I communication and K D2D user equipment (DUE) pairs performing V2V communication, as shown in Figure 1. To increase spectral efficiency, the DUE pairs reuse the orthogonal resource blocks assigned to the CUEs. Time is divided into slots of length T, and slots are grouped into blocks.

The large-scale fading components of the channels, i.e., path loss and shadowing, vary slowly; they are assumed constant within a block and are periodically reported to the BS. The small-scale fading varies from slot to slot in the high-mobility vehicular environment,

so its realization is not known at the BS, which only knows its statistical distribution: unit-mean exponential power gains (Rayleigh fading). In Section 4.5 we also consider errors in the reported large-scale gains.

2.1. Problem Formulation

The channel power gain between CUE m and the BS is $h_m^c = \alpha_m^c g_m^c$, where α_m^c is the slowly varying large-scale fading and g_m^c is the fast varying small-scale fading. The channel of DUE pair k, $h_k^d = \alpha_k^d g_k^d$, the interfering channel from CUE m to the receiver of DUE pair k, $h_{m,k} = \alpha_{m,k} g_{m,k}$, and the interfering channel from the transmitter of DUE pair k to the BS, $h_{k,B} = \alpha_{k,B} g_{k,B}$, are defined similarly. The spectrum of each CUE can be shared by at most one DUE pair, and $x_{m,k} \in \{0,1\}$ is the spectrum allocation indicator: $x_{m,k} = 1$ means that DUE pair k reuses the spectrum of CUE m.

CUEs always have data to transmit. The received SINR at DUE k is

$$\gamma_k^d = \frac{P_k^d \alpha_k^d g_k^d}{\sigma^2 + \sum_{m=1}^{M} x_{m,k} P_m^c \alpha_{m,k} g_{m,k}}, \tag{1}$$

and the received SINR of CUE m at the BS is

$$\gamma_m^c = \frac{P_m^c \alpha_m^c g_m^c}{\sigma^2 + \sum_{k=1}^{K} x_{m,k} P_k^d \alpha_{k,B} g_{k,B}}, \tag{2}$$

where σ^2 is the noise power, and P_k^d and P_m^c are the transmit powers of DUE k and CUE m. Even if CUE m and DUE k share the spectrum, the interference is not persistent, because DUE k does not transmit in a slot if its queue is empty at the beginning of the slot. Let ρ_k^d be the probability that the queue is non-empty at the beginning of a slot. The ergodic capacity of CUE m (in bit/s/Hz) is then

$$R_m^c = \sum_{k=1}^{K} x_{m,k} \left\{ \left(1 - \rho_k^d\right) \mathbb{E}\left[\log_2\left(1 + \frac{P_m^c \alpha_m^c g_m^c}{\sigma^2}\right)\right] + \rho_k^d \, \mathbb{E}\left[\log_2\left(1 + \frac{P_m^c \alpha_m^c g_m^c}{\sigma^2 + P_k^d \alpha_{k,B} g_{k,B}}\right)\right] \right\}. \tag{3}$$

We maximize the sum ergodic capacity of the CUEs while ensuring the latency of each DUE and a minimum capacity R_0 of each CUE. Following [13], the resource-allocation problem is

$$\max_{\{P_k^d\},\{P_m^c\},\{x_{m,k}\}} \sum_{m=1}^{M} R_m^c \tag{4a}$$

$$\text{s.t. } \mu_k^d \le \mu_0, \quad \forall k, \tag{4b}$$

$$R_m^c \ge R_0, \quad \forall m, \tag{4c}$$

$$0 \le P_k^d \le P_{\max}^d, \quad \forall k, \tag{4d}$$

$$0 \le P_m^c \le P_{\max}^c, \quad \forall m, \tag{4e}$$

$$x_{m,k} \in \{0,1\}, \ \textstyle\sum_m x_{m,k} \le 1, \ \sum_k x_{m,k} \le 1, \quad \forall m,k, \tag{4f}$$

where μ_k^d is the average packet sojourn time of DUE k, i.e., the sum of the queueing time and the service (transmission) time, μ_0 is its maximum allowed value, R_0 is the minimum capacity requirement of a CUE, and $P_{\max}^c$ and $P_{\max}^d$ are the maximum transmit powers of CUEs and DUEs.

2.2. Power Allocation for CUE–DUE Pairs

As in [12,13], the power allocation is solved for every possible CUE–DUE reuse pair. Packets arrive at DUE k according to a Poisson process with rate λ_k (packets/s); in every slot, the head-of-line packet is transmitted and is retransmitted in the next slot if the transmission fails [13]. The probability q_k that a transmission fails is

$$q_k = \Pr\left\{ \frac{P_k^d \alpha_k^d g_k^d}{\sigma^2 + P_m^c \alpha_{m,k} g_{m,k}} < \gamma_0 \right\} = 1 - \frac{P_k^d \alpha_k^d \exp\left(-\frac{\gamma_0 \sigma^2}{P_k^d \alpha_k^d} \right)}{P_k^d \alpha_k^d + \gamma_0 P_m^c \alpha_{m,k}}, \tag{5}$$

where γ_0 is the SINR threshold, $\rho_k^d = \lambda_k T/(1-q_k)$, and the average packet sojourn time is

$$\mu_k^d = \frac{T}{2} + \frac{T}{1-q_k} + \frac{\lambda_k T^2 (1+q_k)}{2(1-q_k)(1-q_k-\lambda_k T)}. \tag{6}$$

The three terms are, respectively, the mean wait until the next slot boundary, the mean service time including retransmissions, and the mean queueing delay behind earlier packets. Since μ_k^d increases with q_k, the latency constraint (4b) is satisfied if $q_k \le \bar{q}_k$, where $\bar{q}_k$ is the root of $\mu_k^d = \mu_0$:

$$\bar{q}_k = \frac{2\lambda_k T^2 - (2\lambda_k \mu_0 + 3)T + 2\mu_0}{2\mu_0 - T}. \tag{7}$$

Remark: With the parameters of Table 2 ($T = 0.2$ ms, $\mu_0 = 1$ ms), (7) gives $\bar{q}_k = 0.387$ at $\lambda_k = 2200$ packets/s, 0.244 at 3000 packets/s and 0.031 at 4200 packets/s. A bound on the mean sojourn time therefore tolerates a large per-slot failure probability at moderate loads, and it gives no guarantee on the delay tail. We therefore also report the delay distribution (Section 4.4) and, in one experiment, a stricter reliability-oriented bound: requiring that a packet needs more than $N = \lfloor D_{\max}/T \rfloor$ transmissions with probability at most ϵ gives $q_k^N \le \epsilon$, i.e., $\bar{q}_k^{\mathrm{rel}} = \min\{\bar{q}_k, \epsilon^{1/N}\}$. With $\epsilon = 10^{-5}$ and $D_{\max} = 0.5$ ms, $\bar{q}_k^{\mathrm{rel}} = 3.16 \times 10^{-3}$. This bound neglects queueing, so its effect on the actual delay is evaluated by simulation.

The power allocation problem for each CUE–DUE pair is given by [13]:

$$\max_{P_k^d, P_m^c} \; R_m^c\left(P_k^d, P_m^c\right) \quad \text{s.t.} \quad q_k\left(P_k^d, P_m^c\right) \le \bar{q}_k, \quad 0 \le P_k^d \le P_{\max}^d, \quad 0 \le P_m^c \le P_{\max}^c, \tag{8}$$

where, under Rayleigh fading, the capacity of CUE m paired with DUE k is

$$R_m^c = \left(1 - \rho_k^d\right) \frac{\mathrm{e}^{1/a} E_1(1/a)}{\ln 2} + \rho_k^d \frac{a\left[\mathrm{e}^{1/a} E_1(1/a) - \mathrm{e}^{1/b} E_1(1/b)\right]}{(a-b)\ln 2}, \tag{9}$$

with $a = P_m^c \alpha_m^c / \sigma^2$, $b = P_k^d \alpha_{k,B} / \sigma^2$ and $E_1(x) = \int_x^\infty \mathrm{e}^{-\zeta}/\zeta \, \mathrm{d}\zeta$.

Model-based benchmark: The iterative algorithm of [13] solves (8) using (5) and (9). We have implemented it as a nested one-dimensional search (bisection for the minimum feasible DUE power, and grid search refined by golden-section search over each power); on random test pairs it matched or exceeded an exhaustive 250×250 grid search. Because this solution is optimal for the assumed model and continuous powers, no scheme that uses the same information and a discrete power set can exceed it when the model is exact.

Table 2. Simulation parameters.

Parameter	Value
Number of CUEs M/DUE pairs K	4/4 (4 to 20 in Section 4.3)
Carrier frequency; bandwidth per resource block	2 GHz; 1 MHz
Maximum transmit power of CUE and DUE	23 dBm
Noise power; BS/vehicle antenna gain; noise figure	-114 dBm; 8/3 dBi; 5/9 dB
V2I/V2V path loss	$128.1 + 37.6 \log_{10} d$ (d in km)/WINNER+ B1
Shadowing std. (V2I/V2V); small-scale fading	8 dB/3 dB, log-normal; Rayleigh
Vehicle average speed	60 km/h (60–140 km/h in Figure 3)
Time slot length, T	0.2 ms
Packet arrival rate, λ_k	2200–4200 packets/s (3000 unless stated)
Minimum required capacity, R_0	0.5 Mbps (0.5 bit/s/Hz over 1 MHz)
Maximum average packet sojourn time, μ_0	1 ms
Minimum SINR threshold for DUE, γ_0	5 dB
DQN: hidden layers; activation	520, 280, 140 neurons; ReLU
DQN: optimizer; learning rate; mini-batch	Adam; 5×10^{-4}; 128
DQN: episodes; steps per episode; discount γ	6000; 5; 0.1
DQN: exploration ϵ	1 decaying exponentially to 0.024 over 80% of the episodes
DQN: replay memory; target update	10^5 transitions; every 200 updates
DDPG: actor/critic hidden layers	256, 128 (tanh output)/256, 128
DDPG: learning rates; soft update τ	Adam, 10^{-4} (actor), 10^{-3} (critic); 0.005
DDPG: exploration noise	Gaussian, std. $\max(0.05, 0.6\epsilon)$ on the normalized action
DDPG: mini-batch; replay; γ; episodes	128; 10^5; 0.1; 6000 (same as DQN)
Measurement window, N_{m}	4000 slots
Independent training runs	3 (DQN and DDPG, also at each CSI-error level); 2 (fixed-CUE-power DQN)
Test drops per point	250 (16 candidate pairs each), identical for all schemes

2.3. Spectrum Allocation for CUE–DUE Pairs

Solving (8) gives the power allocation of every possible reuse pair. Since one DUE can share the spectrum of only one CUE, and the capacity of a CUE may fall below R_0, the reuse pairs must be selected. The weight of pair (m, k) is

$$R^*_{m,k} = \begin{cases} R^c_m\left(P^{d*}_k, P^{c*}_m\right), & \text{if (8) is feasible and } R^c_m\left(P^{d*}_k, P^{c*}_m\right) \geq R_0, \\ -\infty, & \text{otherwise,} \end{cases} \tag{10}$$

and the Hungarian method [40], a weighted bipartite matching algorithm with polynomial complexity, returns the pairing with the highest total capacity. A DUE pair whose best available weight is $-\infty$ is not admitted, and its CUE then transmits without interference.

3. Reinforcement Learning for Resource Optimization

RL is a sequential decision-making framework in which an agent interacts with an environment to maximize a numerical reward, and is commonly formulated as a Markov decision process (MDP). At each step t the agent observes a state s_t, takes an action a_t according to a policy π, receives a reward r_t and moves to the next state s_{t+1}, with a probability that depends only on s_t and a_t; the interaction is summarized by the experience $e^{(t)} = (s^{(t)}, a^{(t)}, r^{(t)}, s^{(t+1)})$ [41]. Q-learning learns the action-value function $Q^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{i\geq 0} \gamma^i r_{t+i} \mid s_t = s, a_t = a\right]$, where $\gamma \in [0,1)$ is the discount factor, and the optimal action-value function satisfies the Bellman optimality equation $Q^*(s,a) = \mathbb{E}\left[r_t + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a\right]$ [42]. Tabular Q-learning becomes impractical when the state space is large or continuous, because most states are rarely visited. A deep Q-network (DQN) instead approximates the action-value function with a neural network, $Q(s,a;\boldsymbol{\theta}) \approx Q^*(s,a)$, where $\boldsymbol{\theta}$ are its weights [42,43].

3.1. Proposed DQN Approach

Each candidate CUE–DUE pair is treated as an agent, and all agents share one DQN. In each block, every pair observes its state, selects a power combination as its action, and receives a reward based on the outcome of that action. The elements of the decision process are as follows.

- *Step and episode:* a step is one decision interval within a block; an episode is one block (one vehicle drop) of five steps.
- *State s:* the large-scale information of the pair and the operating conditions (Section 3.1.1).
- *Action a:* a pair of discrete CUE and DUE power levels (Section 3.1.2).
- *Reward r:* computed from the measured outcome of the action (Section 3.1.3).
- *Transition:* within an episode, the large-scale state does not change and is independent of the action, so $s_{t+1} = s_t$; only the small-scale fading, and hence the reward, changes from step to step.

Because the action does not influence the next state, the decision process can be considered to be a contextual bandit, i.e., an MDP whose transitions do not depend on the actions. The optimal action in each state is then the one that maximizes the expected immediate reward $\bar{r}(s,a) = \mathbb{E}[r \mid s,a]$. With $s_{t+1} = s_t$, the Bellman optimality equation gives $V^*(s) = \max_{a'} \bar{r}(s,a') + \gamma V^*(s)$, i.e., $V^*(s) = \max_{a'} \bar{r}(s,a')/(1-\gamma)$, and therefore

$$Q^*(s,a) = \bar{r}(s,a) + \frac{\gamma}{1-\gamma} \max_{a'} \bar{r}(s,a'). \tag{11}$$

The second term of (11) does not depend on a, so $\arg\max_a Q^*(s,a) = \arg\max_a \bar{r}(s,a)$ and any discount factor $\gamma < 1$ leads to the same greedy policy. We nevertheless retain the standard DQN (experience replay, a target network, and a small discount factor $\gamma = 0.1$) for two reasons. First, it stabilizes learning from noisy measured rewards, and second, it allows a later extension in which the queue state is included, so that actions do affect the next state. A small γ keeps the target close to the immediate reward and limits the propagation of estimation errors. As a check, we retrained the DQN with $\gamma = 0$; the resulting sum capacity was between 0% and 4% lower than with $\gamma = 0.1$ across the tested loads, which is within the spread between independent training runs.

Meaning of model-free: The DQN is trained in a simulated environment, which of course contains a channel model. By model-free we mean that the agent never evaluates the analytical expressions (5), (6) or (9), nor the assumed fading distribution. Its rewards are computed from measured transmission outcomes (the observed failure frequency and achieved rate), and at deployment it needs neither the optimization model nor an iterative solver. The learned policy is therefore tied to the conditions under which it was trained, not to the analytical model; if these conditions differ from the analytical model, as in Section 4.5, the policy adapts to the actual conditions while the model-based solution does not.

We use experience replay to train the DQN: the experience tuples are stored in a replay memory, from which mini-batches are sampled at random. Random sampling removes the strong correlation between consecutive experiences, which would otherwise bias learning. The DQN is first trained over many episodes; the trained network is then used to find the power allocation of all CUE–DUE pairs. The state, action, and reward are described below.

3.1.1. State Space

The state of pair (m,k) is the seven-dimensional vector

$$s_{m,k} = [\tilde{\alpha}^c_m,\ \tilde{\alpha}^d_k,\ \tilde{\alpha}_{m,k},\ \tilde{\alpha}_{k,B},\ \lambda_k T,\ \log_{10} \bar{q}_k,\ C], \tag{12}$$

where $\tilde{\alpha} = 10 \log_{10}(P_{\max} \alpha / \sigma^2)$ is each large-scale gain expressed as a signal-to-noise ratio at maximum power (in dB), $\bar{q}_k$ is the latency bound (7) and C is the preference weight discussed in Section 4.2. Each entry is shifted and scaled to be of order one. Only large-scale information is used, so the signaling requirement is the same as that of [12,13].

3.1.2. Action Space

Although transmit powers are continuous, we consider $n = 10$ discrete power levels for both V2V and V2I links, uniformly spaced on a dB scale between -20 dBm and $P_{\max} = 23$ dBm. The action space therefore has $n \times n = 100$ elements. The choice of n trades power-control resolution against the size of the action space: the number of Q-values to learn grows as n^2, whereas the loss due to quantisation decreases with n. With exact-model exhaustive search over the power grid, $n = 5$, 10, 20 and 40 levels attain 96.9%, 98.9%, 99.5% and 99.8% of the continuous optimum at $\lambda = 3000$ packets/s, so going from 10 to 20 levels recovers at most about 0.6 percentage points while multiplying the number of actions by four. The learned DQN with $n = 10$ attains 94.2%; its learning loss already exceeds the quantisation loss, so a finer grid would not help without a larger training budget.

3.1.3. Reward Design

We consider two objectives: maximizing the CUE capacity and reducing the V2V latency. Because the capacity and the failure probability have different units and scales, both are normalized before they are combined. In each step the pair measures, over N_{m} slots, the failure frequency $\hat{q}_k$ of its V2V transmissions and the average achieved V2I rate $\hat{R}_m$. The capacity is normalized by the interference-free capacity of the CUE at maximum power, $R_m^{\max} = \mathbb{E}[\log_2(1 + P_{\max} \alpha_m^c g_m^c / \sigma^2)]$, and the failure frequency by its bound, so that both normalized terms lie in $[0, 1]$ when the latency constraint holds. The reward is

$$r = \begin{cases} C \, \dfrac{\hat{R}_m}{R_m^{\max}} - (1 - C) \, \dfrac{\hat{q}_k}{\bar{q}_k}, & \text{if } \hat{q}_k \le \bar{q}_k, \\ -1.5, & \text{otherwise,} \end{cases} \tag{13}$$

where $C \in [0, 1]$ is the priority of the capacity objective and $1 - C$ that of the latency objective. Since μ_k^d in (6) increases monotonically with q_k, lowering q_k lowers the mean sojourn time; the latency objective is pursued through q_k, and its effect in milliseconds is reported in Section 4.2. The penalty -1.5 is below every feasible reward, so infeasible actions are never preferred; it is a design parameter, and any value below -1, the smallest feasible reward, ranks infeasible actions below all feasible ones.

3.2. Single- and Multi-Objective DQN

The single-objective DQN (SO-DQN) is the case $C = 1$ of (13): it maximizes the CUE capacity, and latency enters only through the constraint. The multi-objective DQN (MO-DQN) uses $C < 1$ and rewards a lower failure probability, i.e., a lower latency than required. Instead of training one network per value of C, we include C in the state and sample it during training ($C = 1$ with probability 0.4, otherwise $C \sim \mathcal{U}[0, 1]$), so that one network provides every trade-off [44,45]. The operator can then select C at runtime according to the service requirements.

3.3. Training the DQN

In each episode, a new vehicle drop is generated with a random average speed in $[60, 140]$ km/h, a random arrival rate $\lambda \in [2000, 4250]$ packets/s, a constraint type (the mean-delay bound (7) or the reliability bound of the Remark, each with probability 0.5)

and a preference weight C. The $M \times K = 16$ candidate CUE–DUE pairs of the drop act as parallel agents for five steps. In each step, the small-scale fading is redrawn, and $N_{\mathrm{m}} = 4000$ slots are used to measure the reward. The experiences are stored in a replay memory, and after every step a mini-batch is used to minimize the squared temporal-difference error:

$$L(\boldsymbol{\theta}) = \sum \left[r^{(t)} + \gamma \max_{a'} Q\left(s^{(t+1)}, a'; \boldsymbol{\theta}^{-}\right) - Q\left(s^{(t)}, a^{(t)}; \boldsymbol{\theta}\right) \right]^2, \tag{14}$$

where $\boldsymbol{\theta}^{-}$ are the weights of a target network, which are copied from the prediction network at regular intervals; using two networks makes learning more stable [43]. Training is summarized in Algorithm 1, and all hyperparameters are listed in Table 2.

Algorithm 1 DQN training

1: Initialize the prediction network with random weights $\boldsymbol{\theta}$, set $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$, and initialize the replay memory
2: **for** each episode **do**
3: Generate a vehicle drop (large-scale fading), and set λ, the constraint type, and C
4: **for** each step $t = 1, \ldots, 5$ **do**
5: Each pair observes its state and selects an action by the ϵ-greedy rule
6: Transmit with the selected powers over N_{m} slots (new small-scale fading) and measure $\hat{q}_k$ and $\hat{R}_m$
7: Compute the rewards (13) and store the experiences in the replay memory
8: Sample a mini-batch and update $\boldsymbol{\theta}$ by minimizing (14)
9: Every 200 updates, set $\boldsymbol{\theta}^{-} \leftarrow \boldsymbol{\theta}$
10: **end for**
11: Decay ϵ
12: **end for**

3.4. Continuous-Action Variant (DDPG)

The formulation above does not depend on the learner. As a continuous-action alternative, we also train a DDPG agent [46] with the same state, reward (13), measurement-based training, and preference conditioning. Its actor network maps the state to the two transmit powers on a continuous dB scale in $[-20, 23]$ dBm through a tanh output layer, so no power quantisation is needed; its critic network estimates the action value $Q(s, a)$ of the state–action pair and is trained with the same temporal-difference target as (14), while the actor is updated along the gradient of the critic with respect to the action. Target networks are updated softly, and exploration adds Gaussian noise to the actor output. We refer to the single- and multi-objective versions as SO-DDPG ($C = 1$) and MO-DDPG ($C < 1$). The hyperparameters are listed in Table 2.

3.5. Implementation Stage

After training, exploration is switched off. Each CUE–DUE pair feeds its state to the trained network and selects the action with the largest Q-value (for DDPG, the action output by the actor). The BS evaluates each candidate pairing from the measured rate and failure frequency and uses the Hungarian method to select the CUE–DUE pairs that maximize the system throughput. The online computation per block is one forward pass per candidate pair, executed as a single batched matrix operation for all $M \times K$ pairs, plus the Hungarian method, which is common to all schemes. The DQN has about 2.0×10^5 weights, i.e., about 0.4 MFLOP per pair; the DDPG actor has about 3.5×10^4 weights, because it outputs two values instead of 100.

4. Numerical Simulations

The vehicular environment follows the freeway case of 3GPP TR 36.885 [47], as in [12,13]: three lanes in each direction, a lane width of 4 m, and a BS located 35 m from the center of the freeway. Vehicles are dropped on a 1 km segment around the BS with exponentially distributed spacing of mean 2.5 s $\times$ v, where v is the average speed. CUEs and DUE transmitters are chosen at random, and the receiver of each DUE pair is the nearest other vehicle. The main parameters are given in Table 2.

Compared schemes: We compared the following schemes through numerical simulations. (i) The model-based optimum of [13]. (ii) An exhaustive search over the same 10×10 power grid using the exact model, which separates the quantisation loss from the learning loss. (iii) The proposed SO-DQN and MO-DQN. (iv) The proposed SO-DDPG and MO-DDPG (Section 3.4), with continuous CUE and DUE powers, which also represent the actor-critic approaches of [30,31]. (v) A DQN in which CUEs transmit at maximum power and only the DUE power is learned, as assumed in [27,28]. All schemes use the Hungarian method for pairing.

Training and test data: Training and test drops are generated independently with different random seeds, so test drops are unseen during training. The same test drops and channel realizations are used for all schemes. Packet arrivals are Poisson. Each learning agent is trained once over the whole range of speeds, loads, constraint types, and preference weights, and the same networks are used in all experiments, except the CSI-error experiment, for which agents are trained at each error level. Performance is always evaluated with the exact expressions (5)–(9), regardless of how each scheme made its decisions. Error bars show $\pm$ one standard deviation across independent training runs for learned schemes and 95% confidence intervals across test drops for the model-based schemes. Capacities are in bit/s/Hz.

Implementation. All schemes are implemented in Python 3.12 with NumPy 2.4 and SciPy 1.17 and run on one core of an Intel Xeon processor at 2.1 GHz; no GPU is used. Training a DQN or a DDPG agent takes about 8 min.

4.1. Sum Capacity Performance of CUE

Figure 2 shows the CUE sum capacity against the packet arrival rate λ_k, under the mean-delay constraint (a) and the stricter reliability bound of the Remark (b); panels (c) and (d) show the percentage of V2V links that are admitted and meet their constraint. A higher arrival rate tightens $\bar{q}_k$ and increases the DUE activity factor, so all schemes lose capacity. Under the mean-delay constraint, the SO-DQN attains 90–95% of the model-based optimum, and under the reliability bound, 85–93%. As expected, it does not exceed the optimum: the exhaustive search over the same 10 levels attains 96–99% and 95–98%, so most of the gap is learning loss rather than quantisation loss, including the margin the agent learns because its rewards are noisy measurements. The SO-DDPG agent, which has no quantisation loss, attains 96–99% and 90–95%, i.e., it is closer to the optimum than the SO-DQN under both constraints and is also much less variable across training runs (Section 4.6). The DQN with fixed CUE power performs well under the loose mean-delay constraint but, under the reliability bound, serves only about 93% of the V2V links within their constraint (Figure 2d); its apparently high capacity comes from rejecting V2V links. This confirms that joint V2I–V2V power control is needed when the V2V constraint is tight.

Figure 3 plots the sum capacity against the average vehicle speed. Higher speeds increase the inter-vehicle spacing and hence the V2V link distance, so the DUEs need more power and cause more interference at the BS; the sum capacity therefore decreases with speed. The same trained networks are used at all speeds, and the SO-DQN stays within 89.3–94.4% of the model-based optimum and the SO-DDPG within 92.7–99.0%.

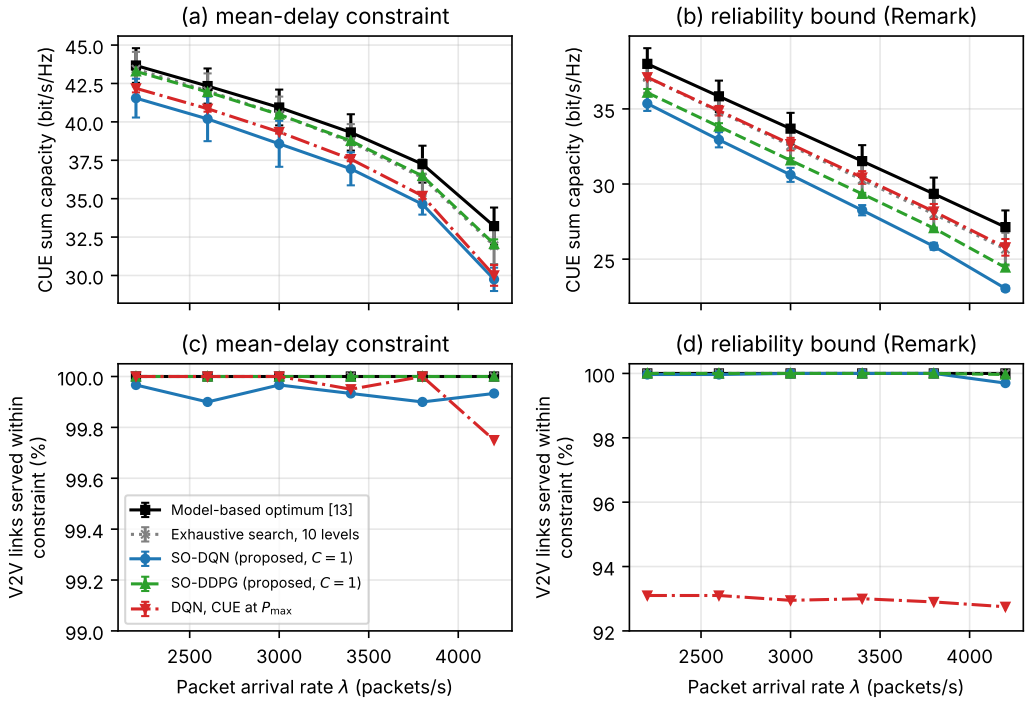


Figure 2. CUE sum capacity (**a**,**b**) and percentage of V2V links admitted and meeting their constraint (**c**,**d**) versus packet arrival rate ($M = K = 4$, $v = 60$ km/h). (**a**,**c**): mean-delay constraint (7); (**b**,**d**): reliability bound of the Remark in Section 2.

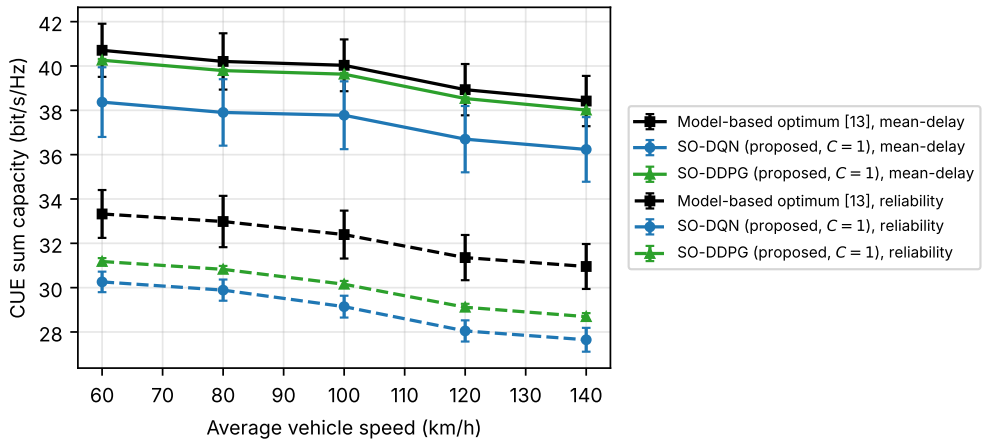


Figure 3. CUE sum capacity versus average vehicle speed ($\lambda = 3000$ packets/s). Solid: mean-delay constraint; dashed: reliability bound.

4.2. Performance Analysis for Multi-Objective Agents

Figure 4 shows the effect of the preference weight C at $\lambda = 3000$ packets/s; all points of each learner are obtained from the same trained network by changing the input C. In panel (b), the latency metric is the mean V2V packet sojourn time μ_k^d of (6) in ms, averaged over the admitted links. As C decreases from 1 (SO-DQN) to 0.2, the mean sojourn time of the DQN falls from 0.55 to 0.45 ms and the mean failure probability from 0.076 to 0.0055, while the sum capacity falls from 38.2 to 30.2 bit/s/Hz. The DDPG agent shows the same behavior: its mean sojourn time falls from 0.66 to 0.45 ms and its capacity from 40.2 to 31.0 bit/s/Hz. The mean sojourn time cannot fall below about 0.45 ms, the value of (6) for error-free links at this load. For $C < 0.2$ the capacity drops sharply with almost no latency gain, so $C \in [0.2, 1]$ is the useful range. Panel (c) shows that the model-based optimum operates exactly on the constraint boundary ($q_k \approx \bar{q}_k$), which maximizes capacity

but leaves no latency margin, whereas the multi-objective agents let the operator trade capacity for latency. For a given mean failure probability, MO-DDPG achieves a higher capacity than MO-DQN under both constraints (e.g., 37.0 versus 35.7 bit/s/Hz at $q_k \approx 0.02$ under the mean-delay constraint), so the preference conditioning works equally well with both learners, and DDPG provides the better trade-off in this nominal setting.

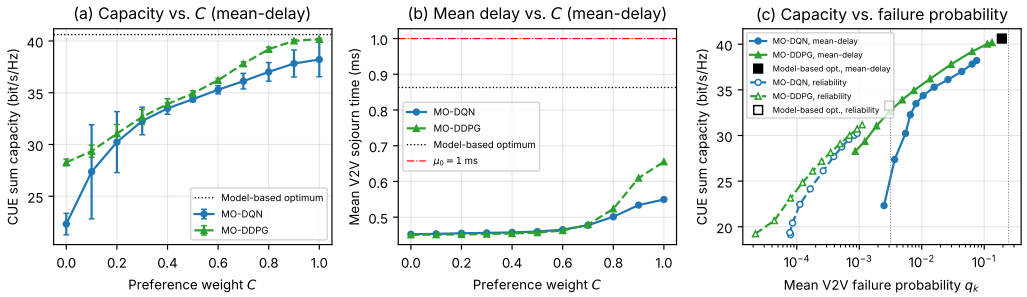


Figure 4. Multi-objective DQN and DDPG ($C = 1$ gives the single-objective agents), $\lambda = 3000$ packets/s. (**a**) Sum capacity and (**b**) mean V2V sojourn time versus C under the mean-delay constraint; error bars: $\pm$ one standard deviation over three runs. (**c**) Sum capacity versus mean V2V failure probability for $C \in \{0, 0.1, \ldots, 1\}$ under both constraints; vertical dotted lines mark the bounds.

4.3. Sum Capacity and Computation Time with Varying Number of Users

Figure 5 varies the numbers of CUEs and DUE pairs together ($M = K$ from 4 to 20) under the mean-delay constraint. Panel (a) shows the average capacity per CUE. The agents trained with $M = K = 4$ are applied without retraining, because they act on one candidate pair at a time; the DQN stays within 94% and the DDPG agent within 99% of the model-based optimum at all sizes. This generalization is expected in the present model, since each resource block is shared by at most one DUE pair and the candidate pairs do not interfere with each other; settings with several DUEs per resource block or inter-cell interference would require retraining or a multi-agent design. Panel (b) shows the online decision time per block, defined as the time to compute the powers of all $M \times K$ candidate pairs plus the Hungarian method, with model loading excluded; the learned agents are timed over 10 repetitions and the model-based search once on each of 6 drops, and the median is reported. The DQN needs 0.23–3.3 ms, compared with 300–593 ms for the model-based search (vectorized over pairs in the same NumPy implementation), i.e., a reduction by a factor of 180–1292. The DDPG agent is still faster (0.05–0.7 ms), because its actor is smaller and outputs two values instead of 100 Q-values. Both learned agents are two to three orders of magnitude faster than the model-based search. Therefore, the difference between them is not significant in practice. The offline training cost (about 8 min on one CPU core for either learner) is not included. An optimized implementation of the model-based search would narrow this gap.

4.4. Delay Distribution

The sojourn time of a packet consists of the wait until the next slot boundary, the queueing delay behind earlier packets, and the transmission delay including retransmissions. To evaluate it beyond its mean, we simulated the retransmission queue of every admitted DUE pair slot by slot, with Poisson arrivals and transmission failures drawn with the probability q_k resulting from the selected powers, using about 5.8×10^6 packets per curve. The simulated mean sojourn time agrees with (6) to within 1% (e.g., 0.880 ms simulated versus 0.883 ms from (6) for the model-based optimum). Figure 6 and Table 3 show the results at $\lambda = 3000$ packets/s.

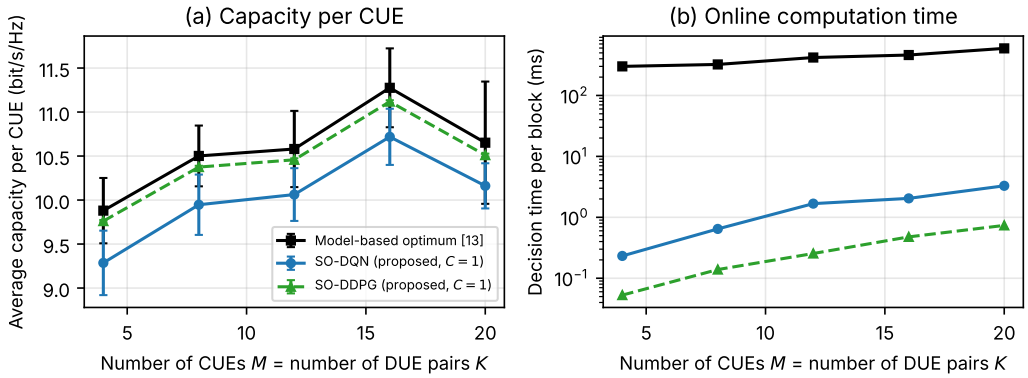


Figure 5. (**a**) Average capacity per CUE and (**b**) online decision time per block versus the number of CUEs M = number of DUE pairs K ($\lambda = 3000$ packets/s, mean-delay constraint).

Under the mean-delay constraint, the model-based optimum meets $\mu_k^d \leq \mu_0$, but 29.5% of the packets exceed 1 ms, because the constraint allows a large failure probability. The SO-DQN reduces this share to 11.3% and the SO-DDPG, which operates closer to the constraint, to 17.2%; the MO-DQN with $C = 0.2$ reduces it to 2.9%. However, even error-free links give $\Pr\{D > 1\,\text{ms}\} = 0.027$ at this load ($\lambda T = 0.6$): the remaining tail is caused by queueing, not by transmission failures. At a light load of $\lambda = 200$ packets/s, the same power decisions give $\Pr\{D > 0.7\,\text{ms}\} = 6.3 \times 10^{-5}$ and no packet above 1 ms among 5.8 million. Power allocation can thus make the link-level contribution to the delay tail small, but it cannot by itself guarantee stringent end-to-end URLLC delay targets at high load; queue-aware scheduling or load control would be required.

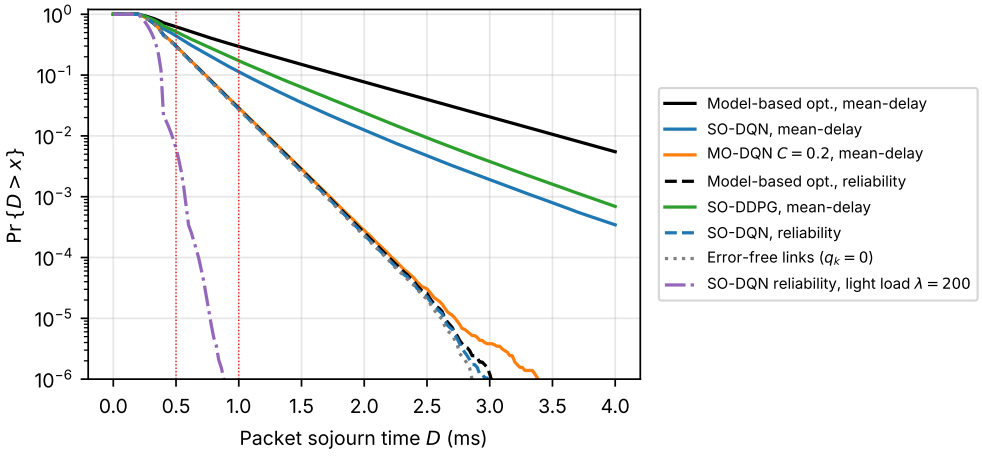


Figure 6. Complementary CDF of the V2V packet sojourn time from queue simulation ($\lambda = 3000$ packets/s unless stated). Vertical lines mark 0.5 ms and 1 ms.

Table 3. V2V delay statistics from queue simulation ($\lambda = 3000$ packets/s unless stated).

Scheme (Constraint)	Mean q_k	Mean Delay (ms)	$\Pr\{D > 1\,\text{ms}\}$	$\Pr\{D > 2\,\text{ms}\}$	99.9th Perc. (ms)
Model-based optimum (mean-delay)	0.199	0.880	0.295	0.077	5.30
SO-DQN (mean-delay)	0.092	0.582	0.113	0.012	3.37
SO-DDPG (mean-delay)	0.138	0.667	0.172	0.024	3.78
MO-DQN, $C = 0.2$ (mean-delay)	4.3×10^{-3}	0.453	0.029	2.8×10^{-4}	1.73
Model-based optimum (rel.)	3.0×10^{-3}	0.452	0.028	2.6×10^{-4}	1.71
SO-DQN (rel.)	1.0×10^{-3}	0.450	0.027	2.4×10^{-4}	1.70
Error-free links ($q_k = 0$)	0	0.449	0.027	2.3×10^{-4}	1.70
SO-DQN (rel.), $\lambda = 200$ packets/s	1.1×10^{-3}	0.304	$<2 \times 10^{-7}$	$<2 \times 10^{-7}$	0.57

4.5. Robustness to Errors in the Large-Scale CSI

The model-based optimum assumes that the reported large-scale gains are exact. In practice, they are estimated, averaged, quantised, and reported with delay. We model this by multiplying each of the four large-scale gains observed by the BS by an independent log-normal error with standard deviation σ_e dB; all schemes observe these noisy values, and performance is evaluated with the true gains. For each σ_e, three DQNs and three DDPG agents were trained with the same level of error, and all schemes were tested on the same 250 drops (1000 admitted V2V links per scheme and run). Figure 7 reports the percentage of admitted V2V links whose true failure probability exceeds the bound, and the sum capacity.

With $\sigma_e = 2$ dB, the model-based optimum violates the constraint on 46.5% (mean-delay) and 56.5% (reliability bound) of the links, because it places each link exactly on a constraint boundary computed from wrong gains; at 4 dB these values are 51.1% and 59.2%. The DQN trained with exact CSI is less affected (0.2% and 1.0% at 2 dB), because the margin it learned from noisy rewards absorbs part of the error; the DDPG agent trained with exact CSI, which operates closer to the boundary, is affected more (0.8% and 2.6%). The agents trained with CSI errors learn the margin that the error requires and reduce the violations to 0.0 ± 0.1% and 0.5 ± 0.3% (DQN) and 0.0 ± 0.0% and 0.2 ± 0.1% (DDPG) at 2 dB and to 0.0 ± 0.0% and 0.3 ± 0.2% (DQN) and 0.0 ± 0.0% and 0.2 ± 0.1% (DDPG) at 4 dB (mean $\pm$ standard deviation over three runs). The price is a lower sum capacity (6.1% below the model-based scheme at 2 dB under the mean-delay constraint for the DQN), which is the cost of actually meeting the constraint. Training under CSI errors, however, affects the two learners differently. Under the mean-delay constraint, the three DQN runs reach similar capacities (37.7 ± 0.6 and 36.1 ± 0.5 bit/s/Hz at 2 and 4 dB, mean $\pm$ standard deviation), whereas the DDPG runs vary widely (35.5 ± 5.9 and 31.7 ± 4.6 bit/s/Hz). Three of the six DDPG runs converged to a policy in which the actor output for the DUE power saturates at its upper limit of 23 dBm. Such a policy satisfies the latency constraint but causes strong interference at the BS and reduces the capacity to about 29 bit/s/Hz. The DDPG runs that did not saturate reached 37.0–39.1 bit/s/Hz, slightly above the DQN. Saturation of a deterministic actor at the edge of its action range is a known difficulty of DDPG. It was triggered here by the noisier rewards and the penalty for infeasible actions, and it does not arise for the DQN, whose action is selected by comparing Q-values over a fixed grid. A model-based design could add a back-off margin, but choosing it requires knowledge of the error statistics, which the learned agents learn from experience.

4.6. Choice of Learner: DQN Versus DDPG

Table 4 summarizes the comparison between the two learners. When the training environment matches the deployment conditions, DDPG is the better choice. It is closer to the model-based optimum because it has no quantisation loss. It also varies much less between training runs. It reaches 90% of its final training reward in slightly fewer episodes and provides a better capacity–latency trade-off. Finally, its decision time is also lower. The DQN does not offer a performance advantage in this setting. Its advantage is training reliability when the rewards are distorted by CSI errors. All six DQN runs converged to similar policies, whereas three of the six DDPG runs saturated at maximum DUE power. In addition, a DQN trained with exact CSI degrades more gracefully when the CSI is inaccurate. Because its coarse power grid and noisy rewards lead it to keep a larger margin from the constraint. We therefore recommend DDPG when an accurate training environment is available, and the DQN, or DDPG with several training runs and a check of the learned policy before deployment, when the training conditions are uncertain. This agrees with a recent multi-agent RL benchmark for V2X, in which actor-critic methods

performed best on the hardest tasks [39]. Both learners use the same formulation. The benefits shown in this paper, namely joint V2I–V2V power control, measurement-based training, preference conditioning and robustness to model mismatch, come from this formulation rather than from the choice of learner.

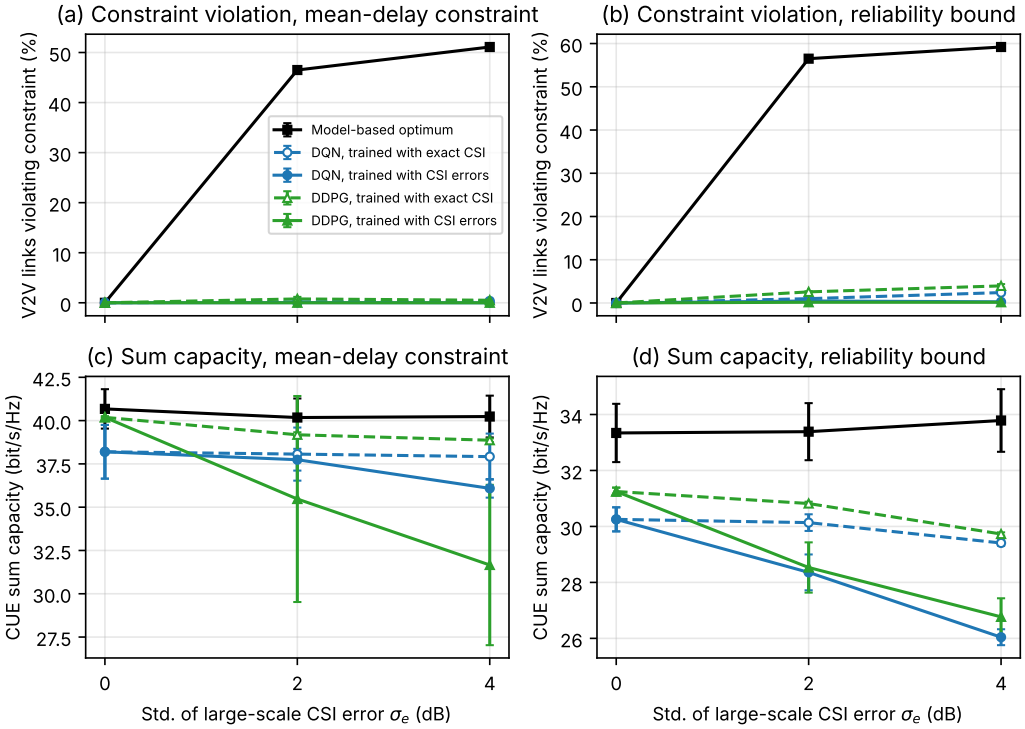


Figure 7. Robustness to large-scale CSI errors ($\lambda = 3000$ packets/s, $M = K = 4$): (**a**,**b**) percentage of admitted V2V links violating the constraint and (**c**,**d**) sum capacity, under the mean-delay constraint (**a**,**c**) and the reliability bound (**b**,**d**). Dashed lines with open markers: agents trained with exact CSI; solid lines: agents trained at the σ_e shown. Error bars: ±one standard deviation over three runs.

Table 4. Comparison of the DQN and DDPG implementations ($\lambda = 3000$ packets/s unless stated; mean ± standard deviation over three training runs).

Criterion	DQN	DDPG
Capacity, % of model-based optimum, mean-delay	$94.2 \pm 3.7\%$	$98.9 \pm 0.1\%$
Capacity, % of model-based optimum, reliability bound	$90.9 \pm 1.4\%$	$93.7 \pm 0.4\%$
Episodes to reach 90% of final training reward	2744	2508
Capacity at $q_k \approx 0.02$, MO agents (bit/s/Hz)	35.7	37.0
Decision time per block, $M = K = 4/20$ (ms)	0.23/3.3	0.05/0.7
Violations at $\sigma_e = 2$ dB, trained with exact CSI (mean-delay/rel.)	0.17/1.00%	0.77/2.57%
Capacity at $\sigma_e = 2/4$ dB, trained with CSI errors, mean-delay (bit/s/Hz)	$37.7 \pm 0.6/36.1 \pm 0.5$	$35.5 \pm 5.9/31.7 \pm 4.6$
Training runs saturating at maximum DUE power (CSI errors)	0 of 6	3 of 6

5. Conclusions

In this paper, a DRL-based scheme was developed that jointly allocates V2I and V2V transmit powers in a D2D-enabled V2X network under a V2V latency constraint, using single- and multi-objective rewards, and was implemented with a DQN and a DDPG agent. Under the nominal channel model, for which the model-based solution of [13] is optimal by construction, the DQN attains 90–95% and the DDPG agent 96–99% of the optimum sum capacity under the mean-delay constraint, at an online computation time two to three orders of magnitude lower. The normalized multi-objective reward allows the operator to trade capacity for latency with a single network. The joint V2I–V2V power control

proved necessary when the V2V constraint is tight. When the reported large-scale gains are inaccurate, agents trained under such conditions keep the V2V constraint satisfied, whereas the model-based solution violates it on about half of the links. DDPG is the better learner when the training environment is accurate, whereas the DQN trains more reliably when the rewards are affected by CSI errors. The study uses Shannon capacity and outage-type failure probabilities.Moreover, at the considered loads, the delay tail is dominated by queueing, which power allocation alone cannot control. Future work will include queue-aware states, finite-blocklength rate models, more robust actor-critic training and distributed operation for NR sidelink Mode 2.

Author Contributions: Conceptualization, F.T. and M.K.; methodology, M.K., S.I.; software, S.I. and S.A.; validation, S.I., A.A.M.; formal analysis, R.T.K. and M.K.; investigation, M.K., S.I. and S.A.; writing F.T., M.K., A.A.M., S.A. and S.I. visualization, M.K.; supervision, R.T.K., M.K. and F.T.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was partly funded by the ICT Division, Government of the People's Republic of Bangladesh. The APC was funded by the University of the West of Scotland.

Data Availability Statement: The simulation code and the data underlying the figures are available from the corresponding author on reasonable request.

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

Abbreviations

The following abbreviations are used in this manuscript:

B5G	Beyond Fifth Generation
BS	Base Station
CSI	Channel State Information
CUE	Cellular User Equipment
D2D	Device to Device
DDPG	Deep Deterministic Policy Gradient
DQN	Deep *Q*-Network
DRL	Deep Reinforcement Learning
DUE	D2D User Equipment
MDP	Markov Decision Process
MO-DQN/SO-DQN	Multi-/Single-Objective DQN
RL	Reinforcement Learning
SINR	Signal-to-Interference-plus-Noise Ratio
URLLC	Ultra-Reliable and Low-Latency Communication
V2I	Vehicle to Infrastructure
V2V	Vehicle to Vehicle
V2X	Vehicle to Everything

References

1. Gao, J.; Khandaker, M.R.A.; Tariq, F.; Wong, K.K.; Khan, R.T. Deep Neural Network Based Resource Allocation for V2X Communications. In *Proceedings of the 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.
2. Li, T.H.; Khandaker, M.R.A.; Tariq, F.; Wong, K.K.; Khan, R.T. Learning the Wireless V2I Channels Using Deep Neural Networks. In *Proceedings of the 2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–5.
3. Tariq, F.; Khandaker, M.R.A.; Wong, K.K.; Imran, M.A.; Bennis, M.; Debbah, M. A Speculative Study on 6G. *IEEE Wirel. Commun.* **2020**, *27*, 118–125. [CrossRef]
4. Garcia, M.H.C.; Molina-Galan, A.; Boban, M.; Gozalvez, J.; Coll-Perales, B.; Şahin, T.; Kousaridas, A. A Tutorial on 5G NR V2X Communications. *IEEE Commun. Surv. Tutorials* **2021**, *23*, 1972–2026. [CrossRef]

5. Parvini, M.; Schulz, P.; Fettweis, G. Resource Allocation in V2X Networks: From Classical Optimization to Machine Learning-Based Solutions. *IEEE Open J. Commun. Soc.* **2024**, *5*, 1958–1974. [CrossRef]
6. Sun, W.; Ström, E.G.; Brännström, F.; Sou, K.C.; Sui, Y. Radio resource management for D2D-based V2V communication. *IEEE Trans. Veh. Technol.* **2015**, *65*, 6636–6650. [CrossRef]
7. Sun, W.; Yuan, D.; Ström, E.G.; Brännström, F. Cluster-based radio resource management for D2D-supported safety-critical V2X communications. *IEEE Trans. Wirel. Commun.* **2015**, *15*, 2756–2769. [CrossRef]
8. Ashraf, M.I.; Liu, C.F.; Bennis, M.; Saad, W.; Hong, C.S. Dynamic resource allocation for optimized latency and reliability in vehicular networks. *IEEE Access* **2018**, *6*, 63843–63858. [CrossRef]
9. Yang, H.; Zhang, K.; Zheng, K.; Qian, Y. Joint Frame Design and Resource Allocation for Ultra-Reliable and Low-Latency Vehicular Networks. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3607–3622. [CrossRef]
10. Liang, L.; Xie, S.; Li, G.Y.; Ding, Z.; Yu, X. Graph-based resource sharing in vehicular communication. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 4579–4592. [CrossRef]
11. Liang, L.; Kim, J.; Jha, S.C.; Sivanesan, K.; Li, G.Y. Spectrum and power allocation for vehicular communications with delayed CSI feedback. *IEEE Wirel. Commun. Lett.* **2017**, *6*, 458–461. [CrossRef]
12. Liang, L.; Li, G.Y.; Xu, W. Resource allocation for D2D-enabled vehicular communications. *IEEE Trans. Commun.* **2017**, *65*, 3186–3197. [CrossRef]
13. Guo, C.; Liang, L.; Li, G.Y. Resource allocation for low-latency vehicular communications with packet retransmission. In *Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM)*; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
14. Guo, C.; Liang, L.; Li, G.Y. Resource Allocation for High-Reliability Low-Latency Vehicular Communications with Packet Retransmission. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6219–6230. [CrossRef]
15. Guo, C.; Liang, L.; Li, G.Y. Resource Allocation for Vehicular Communications with Low Latency and High Reliability. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 3887–3902. [CrossRef]
16. Guo, C.; Liang, L.; Li, G.Y. Resource Allocation for Low-Latency Vehicular Communications: An Effective Capacity Perspective. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 905–917. [CrossRef]
17. Polyanskiy, Y.; Poor, H.V.; Verdú, S. Channel Coding Rate in the Finite Blocklength Regime. *IEEE Trans. Inf. Theory* **2010**, *56*, 2307–2359. [CrossRef]
18. Durisi, G.; Koch, T.; Popovski, P. Toward Massive, Ultrareliable, and Low-Latency Wireless Communication with Short Packets. *Proc. IEEE* **2016**, *104*, 1711–1726. [CrossRef]
19. Khan, N.; Coleri, S. Resource Allocation for Ultra-Reliable Low-Latency Vehicular Networks in Finite Blocklength Regime. In *Proceedings of the 2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*; IEEE: Piscataway, NJ, USA, 2022; pp. 322–327. [CrossRef]
20. Hegde, A.; Song, R.; Festag, A. Radio Resource Allocation in 5G-NR V2X: A Multi-Agent Actor-Critic Based Approach. *IEEE Access* **2023**, *11*, 87225–87244. [CrossRef]
21. Husain, M.I.; Haque, M.E.; Tariq, F. An Efficient Packet Scheduling Algorithm for URLLC Systems. In *Proceedings of the 2020 International Conference on UK-China Emerging Technologies (UCET)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–4.
22. Haque, M.E.; Tariq, F.; Khandaker, M.R.A.; Wong, K.K.; Zhang, Y. A Survey of Scheduling in 5G URLLC and Outlook for Emerging 6G Systems. *IEEE Access* **2023**, *11*, 34372–34396. [CrossRef]
23. Tariq, F.; Khandaker, M.; Ansari, I.S. *6G Wireless: The Communication Paradigm Beyond 2030*; CRC Press: Boca Raton, FL, USA, 2023.
24. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; MIT Press: Cambridge, MA, USA, 2018.
25. Huang, Q.; Xie, X.; Tang, H.; Hong, T.; Kadoch, M.; Nguyen, K.K.; Cheriet, M. Machine-learning-based cognitive spectrum assignment for 5G URLLC applications. *IEEE Netw.* **2019**, *33*, 30–35. [CrossRef]
26. Sun, H.; Chen, X.; Shi, Q.; Hong, M.; Fu, X.; Sidiropoulos, N.D. Learning to optimize: Training deep neural networks for wireless resource management. In *Proceedings of the 2017 IEEE 18th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
27. Liang, L.; Ye, H.; Li, G.Y. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2282–2292. [CrossRef]
28. Ye, H.; Li, G.Y.; Juang, B.H.F. Deep reinforcement learning based resource allocation for V2V communications. *IEEE Trans. Veh. Technol.* **2019**, *68*, 3163–3173. [CrossRef]
29. Vu, H.V.; Liu, Z.; Nguyen, D.H.; Morawski, R.; Le-Ngoc, T. Multi-agent reinforcement learning for joint channel assignment and power allocation in platoon-based C-V2X systems. *arXiv* **2020**, arXiv:2011.04555.
30. Saied, A.; Qiu, D.; Swessi, M. Resource Management Based on Reinforcement Learning for D2D Communication in Cellular Networks. In *Proceedings of the 2020 International Symposium on Networks, Computers and Communications (ISNCC)*; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6.

31. Kasgari, A.T.Z.; Saad, W. Model-free ultra reliable low latency communication (URLLC): A deep reinforcement learning framework. In *Proceedings of the 2019 IEEE International Conference on Communications (ICC)*; IEEE: Piscataway, NJ, USA, 2019; pp. 1–6.
32. Zhang, X.; Peng, M.; Yan, S.; Sun, Y. Deep-reinforcement-learning-based mode selection and resource allocation for cellular V2X communications. *IEEE Internet Things J.* **2019**, *7*, 6380–6391. [CrossRef]
33. Ding, H.; Leung, K.C. Resource Allocation for Low-Latency NOMA-V2X Networks Using Reinforcement Learning. In *Proceedings of the IEEE INFOCOM 2021—IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
34. Yuan, Y.; Zheng, G.; Wong, K.K.; Letaief, K.B. Meta-Reinforcement Learning Based Resource Allocation for Dynamic V2X Communications. *IEEE Trans. Veh. Technol.* **2021**, *70*, 8964–8977. [CrossRef]
35. Sohaib, R.M.; Onireti, O.; Sambo, Y.; Swash, R.; Imran, M. Energy Efficient Resource Allocation Framework Based on Dynamic Meta-Transfer Learning for V2X Communications. *IEEE Trans. Netw. Serv. Manag.* **2024**, *21*, 4343–4356. [CrossRef]
36. Khan, N.; Coleri, S. Event-Triggered Reinforcement Learning Based Joint Resource Allocation for Ultra-Reliable Low-Latency V2X Communications. *IEEE Trans. Veh. Technol.* **2024**, *73*, 16991–17006. [CrossRef]
37. Khan, N.; Abdallah, A.; Celik, A.; Eltawil, A.M.; Coleri, S. Explainable AI-Aided Feature Selection and Model Reduction for DRL-Based V2X Resource Allocation. *IEEE Trans. Commun.* **2025**, *73*, 7633–7649. [CrossRef]
38. Ji, M.; Wu, Q.; Fan, P.; Cheng, N.; Chen, W.; Wang, J.; Letaief, K.B. Graph Neural Networks and Deep Reinforcement Learning Based Resource Allocation for V2X Communications. *arXiv* **2024**, arXiv:2407.06518.
39. Wang, S.; Lei, L.; Maheshwari, P.; Bellefeuille, S.; Zheng, K. Multi-Agent Reinforcement Learning for V2X Resource Allocation: Disentangling MARL Challenges Through Benchmarking. *arXiv* **2026**, arXiv:2603.06607.
40. West, D.B. *Introduction to Graph Theory*; Prentice Hall: Upper Saddle River, NJ, USA, 2001; Volume 2.
41. Nasir, Y.S.; Guo, D. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2239–2250. [CrossRef]
42. Ahmed, K.I.; Hossain, E. A deep q-learning method for downlink power allocation in multi-cell networks. *arXiv* **2019**, arXiv:1904.13032.
43. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjel, A.K.; Ostrovski, G.; et al. Human-Level Control through Deep Reinforcement Learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
44. Abels, A.; Roijers, D.M.; Lenaerts, T.; Nowé, A.; Steckelmacher, D. Dynamic Weights in Multi-Objective Deep Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*; PMLR: Cambridge, MA, USA, 2019; Volume 97, pp. 11–20.
45. Hayes, C.F.; Rădulescu, R.; Bargiacchi, E.; Källström, J.; Macfarlane, M.; Reymond, M.; Verstraeten, T.; Zintgraf, L.M.; Dazeley, R.; Heintz, F.; et al. A Practical Guide to Multi-Objective Reinforcement Learning and Planning. *Auton. Agents Multi-Agent Syst.* **2022**, *36*, 26. [CrossRef]
46. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous Control with Deep Reinforcement Learning. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016 .
47. 3GPP. *Study on LTE-Based V2X Services (Release 14)*; Technical Report TR 36.885; v14.0.0; 3rd Generation Partnership Project: Sophia Antipolis, France, 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.