electronics

MDPI

Article

Multi-Agent DRL-Based Resource Scheduling and Energy Management for Electric Vehicles

Zhewei Zhang [1], Chengbo Yu [1,*] and Bingxin Tian [2]

[1] School of Electrical and Electronic Engineering, Chongqing University of Technology, Chongqing 400054, China; zzw1137289395@163.com
[2] School of Electronic Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; fytian11@outlook.com
* Correspondence: yuchengbo@cqut.edu.cn

Abstract: With the emergence of vehicular edge computing (VEC) and electric vehicles (EVs), integrating computation and charging tasks presents challenges due to limited resources and dynamic vehicular networks. This research focuses on the joint optimization of computation offloading and charging scheduling in VEC networks. Specifically, we optimize the offloading factor, charging association variable, and charging rates to minimize the system delay and energy consumption by leveraging the multi-attributes of EVs in both information and energy networks. Considering the dynamic environment, we model the problem as a Markov Decision Process, and use the Multi-Agent Reinforcement Learning (MARL) algorithm MADDPG, with its centralized training and distributed execution mechanisms. Simulation results demonstrate that this approach significantly improves utility while reducing energy consumption and latency.

Keywords: deep reinforcement learning; multi-agent deep deterministic policy gradient; computation offloading; charging scheduling

check for updates

Citation: Zhang, Z.; Yu, C.; Tian, B. Multi-Agent DRL-Based Resource Scheduling and Energy Management for Electric Vehicles. *Electronics* **2024**, *13*, 3311. https://doi.org/10.3390/electronics13163311

Academic Editor: Fabio Corti

Received: 20 July 2024
Revised: 15 August 2024
Accepted: 16 August 2024
Published: 21 August 2024

1. Introduction

With the advancement of EVs, they are increasingly pivotal in addressing global energy and environmental challenges, promoting sustainable mobility [1]. EVs demand significant charging infrastructure and computational resources, especially for navigation and autonomous driving. Technologies like vehicular networks and mobile edge computing (MEC) have emerged to meet these needs. Vehicular networks enhance road safety and traffic efficiency, while MEC reduces latency and conserves onboard computing energy [2,3]. In addition to MEC, coordinated charging strategies are essential to manage fleet energy demands without overburdening the electrical grid [4]. Despite advancements, computational offloading and energy management have evolved independently, often leading to suboptimal resource use. Our research integrates EV computational offloading and charging scheduling, coupling information and energy domains to ensure optimal resource allocation and enhanced efficiency, leveraging the full potential of EVs in a smart mobility ecosystem.

1.1. Related Work

In recent years, significant advancements have been made in research on objective optimization within vehicular edge computing, particularly using convex and nonconvex optimization techniques [5] to develop data offloading strategies. Dinh et al. proposed a computation offloading framework to reduce energy consumption in mobile devices (MDs) and minimize task execution delays, developing two approximate solutions for the mixed-integer linear programming problem: one using linear relaxation and the other using semidefinite relaxation [6]. Yan et al. explored resource allocation and task offloading with the goal of minimizing the task execution time and energy consumption of MDs.

They employed a bisection search method to address the mixed-integer optimization problem [7]. Moving beyond conventional optimization, machine learning and deep learning have significantly enhanced predictive and decision-support processes in MEC. These technologies use historical data to anticipate computational demands and effectively manage offloading tasks. Lyu et al. improved edge server data partitioning using stochastic gradient descent [8], and Ale et al. devised a deep recurrent neural network aimed at anticipating user requests, thereby supporting decisions for content offloading and resource allocation based on these anticipations [9]. Despite their effectiveness in navigating complex scenarios, these techniques often rely on extensive, labeled datasets, which are expensive and labor intensive, posing significant challenges in dynamic settings.

EVs have become increasingly popular as a clean and efficient means of transportation, contributing to environmental preservation and a low-carbon lifestyle, but the unpredictable nature of human behavior and various ambient factors, such as electricity prices and weather conditions, complicate energy management for EV charging. The efficient scheduling of EV fleet charging is crucial for reducing energy cost and managing peak loads [10]. With the aim of maximizing profits by scheduling charging and discharging strategies, the EV charging scheduling problem is typically addressed as an optimization problem in traditional research. Common approaches include linear programming, mixed-integer linear programming, dynamic programming, and robust optimization methods [11,12]. For example, Satish et al. developed optimization models using mixed-integer linear programming to determine optimal charging schedules [13]. Yao et al. developed a binary programming strategy for real-time scheduling in response to curtailment requests from utilities [14]. Zhao et al. used pricing incentives to better coordinate EV charging [15]. Despite their adequate performance in charging scheduling, these methods heavily rely on accurate models and fully observable environments, limiting their adaptability and effectiveness in real-time, dynamic settings. In summary, existing research on EV charging scheduling focuses on explicit optimization models, which are significantly influenced by model accuracy. These traditional methods often fall short in dynamic and uncertain environments.

Deep reinforcement learning (DRL) has emerged as a powerful alternative to traditional optimization and supervised deep learning, enabling effective operation in uncertain and dynamic environments without pre-labeled data [16]. By integrating reinforcement learning with deep learning, DRL supports real-time adaptable decision-making, which is crucial for optimizing computation offloading in mobile edge computing (MEC) and managing electric vehicle (EV) charging [17]. Unlike traditional methods, DRL continuously learns and adapts to changing environments, offering flexibility and resilience in managing EV charging and resource planning in MEC [18]. Techniques such as deep Q-networks (DQNs) and twin delayed deep deterministic policy gradient (TD3) have shown potential in handling sequential decision-making problems, providing efficient solutions without relying on explicit system models or extensive data labeling [19]. Numerous studies have shown that DRL significantly optimizes task offloading and resource allocation. In mobile edge computing (MEC) and vehicular edge computing networks (VECNs), multi-agent deep reinforcement learning (MARL) methods have also demonstrated great potential. Zhou et al. proposed a distributed Multi-Agent Reinforcement Learning (DMRE) approach to optimize edge caching strategies in vehicular networks [20]. This method reduces redundant content transmission in the system by coordinating the cache between multiple roadside units (RSUs), significantly enhancing cache resource utilization and system performance. Additionally, Zhou et al. explored the potential of federated distributed deep reinforcement learning in recommendation-enabled edge caching. They introduced a federated learning approach combined with distributed deep reinforcement learning to optimize content recommendation and caching strategies in edge computing environments. This method not only improved the cache hit rate at edge nodes but also significantly reduced bandwidth consumption and content transmission delay [21]. Moreover, Zhou et al. developed a novel Deep Reinforcement Learning-based Computation Offloading and Service Caching Mechanism, named DRLCOSCM, to jointly optimize computation offload-

ing, service caching, and resource allocation strategies in a three-tier mobile cloud-edge computing structure. The approach significantly reduces the cost of the Cloud Service Center (CSC) while ensuring the delay requirements of mobile users (MUs) [22]. Liu and Liao introduced an Actor–Critic algorithm combining policy gradients with temporal difference methods to optimize these tasks in Vehicular Edge Computing Networks (VECNs) [23]. Similarly, Song et al. proposed a semi-online computation offloading model based on Dueling DQNs, considering user behavior predictions and server load balancing [24]. In EV charging scheduling, Zhang et al. employed the deep deterministic policy gradient (DDPG) algorithm for continuous EV charging control [25]. F. L. Da Silva et al. used a combination of Q-learning and a multi-agent framework to manage EV charging coordination within a distribution grid [26].

1.2. Contribution and Structure

Despite significant progress, most studies focus on optimizing either computation offloading or charging scheduling independently, without considering joint optimization. This isolated approach overlooks the interplay between the computational and energy domains of EVs. To tackle these challenges, this paper introduces a novel method integrating both the charging and computational attributes of EVs for joint optimization.

To address the identified challenges and research gaps in coordinating vehicular network charging stations and edge computing with EV charging schedules, this paper introduces a novel cross-domain optimization approach. The main contributions of this paper are summarized as follows:

- The cross-network wireless resource allocation and energy management problem is formulated as a mixed-integer nonlinear program, considering the dynamics of EVs and the uncertainty of renewable energy. The objective is to minimize both the total delay and energy consumption of EVs and the total energy consumption of charing stations. This is achieved by optimizing offloading decisions in the information network and charging decisions and rates in the energy network.
- To address this issue, the problem is reframed as a Markov Decision Process (MDP) and a Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is employed. Each EV acts as an agent to jointly optimize EV-assisted charging station edge computing and charging schedules. Through offline training of the MADDPG model, each EV can make real-time charging station association and resource allocation decisions, thereby maximizing overall system utility by leveraging enhanced scheduling exploration and experience sampling strategies.
- Extensive numerical simulations were conducted to validate the effectiveness of the proposed method. The results demonstrate significant improvements in task performance and reductions in energy consumption and delay across various numbers of EVs. Several DRL algorithms were compared, including DDPG under a fully centralized mechanism (FC-DDPG), DDPG under a fully decentralized mechanism (FD-DDPG), and Actor–Critic. The results show that MADDPG outperforms the others in terms of EV energy consumption, delay, and charging station energy consumption, achieving faster convergence rates and better long-term utility.

The rest of this paper is organized as follows. Section 2 outlines our system model, including the communication model, computation model, and EV charging model, along with the problem formulation. In Section 3, we represent the joint optimization problem as an MDP and propose a joint optimization algorithm based on the MADDPG framework. Section 4 provides the numerical results for all algorithms. Lastly, the conclusions are summarized in Section 5.

2. System Model and Problem Formulation

The Vehicle-to-Grid (V2G) system necessitates the consideration of various factors, including vehicle density, location, and speed within the traffic network, to optimize the routing of electric vehicles and minimize energy consumption and time costs. The system

requires the rational allocation of communication, computing, and caching resources of electric vehicles to enhance overall system performance. Regarding the energy network, the effective management of EV battery energy and control of bi-directional energy flow are essential, while taking into account battery life, energy supply and demand equilibrium, and grid stability.

A novel system model for computational offloading is proposed in the V2G systems as shown in Figure 1. This model involves EVs and charging stations, where charging stations offload computational tasks to the EVs during their charging process. The computational tasks offloaded onto the EVs can range from the analysis of charging demands to other complex computations required at the charging station. In this scenario, EVs move at a constant speed V, providing a stable platform for computational offloading. These tasks benefit from the computational power available on board the EVs. The charging station sends computational requests to the EV, which in turn processes the tasks and returns the results back to the charging station. Hence, the EVs can provide computational services to the charging stations while charging, creating a symbiotic relationship that optimizes resources and enhances efficiency within the V2G system. In the energy domain, to guarantee timely recharging for an EV, numerous charging stations are planned to be installed within the EVs' operational zone. Various types of EVs communicate their specific charging requirements and anticipated price during each time slot. The optimal charging price and rate are determined by comparison, after which operators distribute and assign them to the appropriate charging station for recharging. For convenience, Table 1 lists the main notations used in this paper.

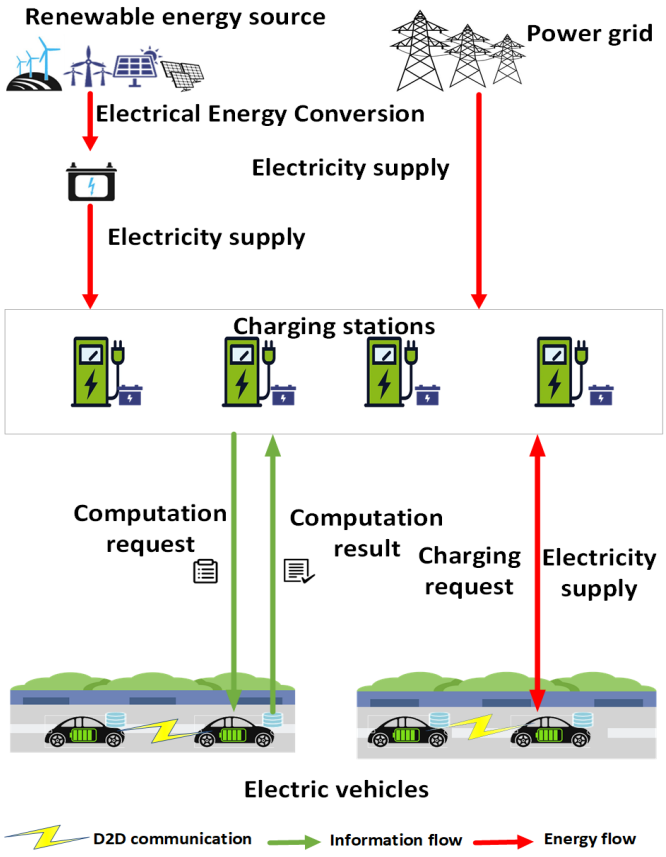


Figure 1. EV computation and charging networks.

Table 1. Symbols and notations.

Symbol	Description	Symbol	Description
K	Total number of charging stations in the V2G system.	N	Maximum number of EVs in the V2G system.
N_0	The spectrum density of the additive white Gaussian noise.	$AL_{k,n}^{(t)}$	Average path loss.
$d_{k,n}^{(t)}$	The Euclidean distance between EV n and charging station k.	ϕ_1	Path loss exponent.
p_k	The transmit power of charging station k.	N	The number of CPU cycles needed to process a single bit of task input.
ψ	Task drop loss	B	The bandwidth of all the channels.
$g_{k,n}^{(t)}$	Channel gain	$F^{(t)}$	The charging station produces computation tasks correlated with a data volume.
$o_{k,n}^{(t)}$	Offloading factor	$G_{n,k}^{(t)}$	The association between the n-th EV and the k-th CS at time slot t.
$\mathbf{p}^{(t)}$	Offloading policy	f_e	CPU frequency for charging station.
$T_{k,l}^{(t)}$	The delay of execution for CS computation at a given time slot t.	$E_{k,l}^{(t)}$	The energy for executing CS computations.
$T_{k,n}^{(t)}$	The delay required to offload data of charging station k for EV computing.	$E_{k,n}^{(t)}$	The energy for the charging station k to offload tasks to EV n at time slot t.
$T_{n,v}^{(t)}$	The delay of execution for n-th EV computation at a given time slot t.	f_v	CPU frequency of the EVs for each cycle.
$E_{n,v}^{(t)}$	Energy for executing EV computations.	$T_{n,v}^{(t)}$	The power required for maintenance when the charging station k is in an idle state.
$E_{k,e}^{(t)}$	The total energy for charging station k for EV computation.	$E_{k,c}^{(t)}$	The total energy for charging station k.
$v_n^{(t)}$	The charging rate of EV n.	$v_k^{(t)}$	The charging rate of charging station k.
$\delta^{(t)}$	The intensity of solar radiation.	$v^{(t)}$	The wind speed.
$X_{k,p}^{(t)}$	The generation rate by the photovoltaic.	$X_{k,p}^{(t)}$	The power generated by the wind power.
$X_k^{(t)}$	The rate of renewable energy production.	$J_k^{(t)}$	The quantity of renewable energy stored in the battery at charging station k.
$H_k^{(t)}$	The rate at which charging station k purchases energy from the grid.	$L_k^{(t)}$	The energy state of the battery for the k-th charging station.
$l_k^{(t)}$	The amount of energy withdrawn from the battery by charging station k.	$W_k^{(t)}$	The demand for charging station k.
$h_n^{(t)}$	The power requirement of the n-th EV.	$\mathcal{T}_n^{(k)}$	The total latency for EV n.

2.1. Communication Model

Consider a V2G system; let $k \in \{1, 2, \ldots, K\}$ represent the k-th charging station, where K is the total number of charging stations, and $n \in \{1, 2, \ldots, N\}$ represent the n-th EV, where N is the maximum number of EVs that the V2G system can receive. The duration of time is consistently segmented into slots, denoted by the index t, with $t \in \{1, 2, \ldots, T\}$.

V2G systems enable the exchange of computational tasks between charging station and electric vehicles. These systems leverage the computational abilities of EVs to perform tasks such as charging demand analysis and other computations. To facilitate efficient task offloading, a communication model is required that considers factors such as signal-to-noise ratio, channel gain, and distance between the charging station and the EV. We present a communication model based on Shannon's formula, incorporating the $B\log_2(1 + SNR)$ term to account for the impact of signal quality on data transmission rates, where B represents the bandwidth and SNR denotes the signal-to-noise ratio. In this model, we

assume constant power levels and neglect interference. The noise (N_0) is considered the only source of disturbance in the channel.

The rate of communication between the charging station and the EV is determined by the distance between them, which directly affects the channel gain. The channel gain represents the attenuation or amplification of the transmitted signal due to the distance-dependent path loss. As the distance increases, the channel gain decreases, resulting in a lower signal quality and reduced data transmission rate. In the context of downlink communications between charging stations and electric vehicles, the implementation of a probabilistic path loss model is utilized, i.e., the average path loss can be given by

$$AL_{k,n}^{(t)} = \phi_1 {d_{n,k}^{(t)}}^{-\alpha}, \tag{1}$$

where $d_{n,k}^{(t)}$ is the Euclidean distance between EV n and charging station k. ϕ_1 is the reference channel gain at one unit distance in meter. α is the path loss exponent.

Let $\gamma_{n,k}^{(t)}$ denote the small-scale fading of the link between charging station k and EV n at time slot t, with $\mathbb{E}\left|\gamma_{n,k}^{(t)}\right|^2 = 1$. The fading factor of the independent random channel adheres to an exponential distribution with a mean of one. Define p_k as the transmit power of charging station k, which is a constant. Therefore, the achievable downlink data rate from charging station k to EV n at time slot t is given by

$$R_{k,n}^{(t)} = B \log_2 \left(1 + \frac{p_k g_{n,k}^{(t)}}{\sigma^2}\right), \tag{2}$$

where B is the bandwidth of all the channels. $g_{k,n}^{(t)} = AL_{n,k}^{(t)}\left|\gamma_{n,k}^{(t)}\right|^2$ is the channel gain. σ^2 is the noise power.

2.2. Computation Model

During the time slot t, the charging station generates computation tasks associated with a data volume of $F^{(t)}$ bits. The execution duration of these tasks does not surpass the length of a given time slot. The offloading factor is denoted by $o_{n,k}^{(t)}$. If $o_{n,k}^{(t)} = 0$, the charging station performs computation tasks locally. Conversely, if $o_{n,k}^{(t)} = 1$, it transfers all computational tasks to EV n. At time slot t, the charging station chooses its offloading policy, represented as $\mathbf{P}^{(t)} = \left[o_{n,k}^{(t)}, G_{n,k}^{(t)}\right]$, from the entire set of potential policies denoted by $\mathbf{P}$.

(1) *CS Computing*: Charging station executes computing tasks locally. The Central Processing Unit of the CSs serves as the principal mechanism for computation. The functionality of the CPU is managed by adjusting its cycle frequency. Local computing for executing $\left(1 - o_{k,n}^{(t)}\right)F^{(t)}$ input bits occurs at the EVs. The variable M is used to represent the quantity of CPU cycles needed for the computation of a single input bit. The bits represented by $\left(1 - o_{k,n}^{(t)}\right)F^{(t)}$ require a total of $\left(1 - o_{k,n}^{(t)}\right)F^{(t)}M$ CPU cycles for processing. Adjusting the CPU frequency $f_{k,e}$ for each cycle e, where $e \in \left\{1, 2, \ldots, (1 - o_{k,n}^{(t)})F^{(t)}M\right\}$, allows the regulation of energy consumption during local task execution at the charging station. The delay of execution for CS computation at a given time slot t can be represented by $T_{k,l}^{(t)}$ as shown in the subsequent formula:

$$T_{k,l}^{(t)} = \sum_{e=1}^{(1-o_{k,n}^{(t)})F^{(t)}M} \frac{1}{f_{k,e}}. \tag{3}$$

At time slot t, the charging station k utilizes $E_{k,l}^{(t)}$ energy for executing local computations, which can be given by:

$$E_{k,l}^{(t)} = \sum_{e=1}^{(1-o_{k,n}^{(t)})F^{(t)}M} \gamma f_{k,e}^2, \tag{4}$$

where the effective capacitance coefficient, denoted as γ, is contingent upon the architecture of the chip.

(2) *EV Computing*: Charging stations offload computational tasks to EV during charging, utilizing the EVs' computational power. Based on the communication model outlined in Section 2.1, the charging station k transmits computational tasks to the EV n at the time slot t using the downlink radio transmission rate $R_{k,n}^{(t)}$. The delay required to offload $o_k^{(t)}F^{(t)}$ bits of data of charging station k is denoted as $T_{k,n}^{(t)}$:

$$T_{k,n}^{(t)} = \frac{o_{k,n}^{(t)}F^{(t)}}{R_{k,n}^{(t)}}. \tag{5}$$

The energy $E_{k,n}^{(t)}$ expended by the charging station to offload tasks to EV n at time slot t is reliant on the power p_k used for offloading, and the transmission duration $T_{k,n}^{(t)}$ is given by:

$$E_{k,n}^{(t)} = \frac{o_{k,n}^{(t)}F^{(t)}p_k}{R_{k,n}^{(t)}}. \tag{6}$$

We can also define the computational energy consumption and delay of EVs. The delay of execution for n-th EV computation at a given time slot t can be represented by $T_{n,v}^{(t)}$ as shown in the subsequent formula:

$$T_{n,v}^{(t)} = \sum_{v=1}^{o_{k,n}^{(t)}F^{(t)}M} \frac{1}{f_{n,v}}, \tag{7}$$

where the $f_{n,v}$ is the the CPU frequency of the EVs for each cycle v, and $v \in \left\{1, 2, \ldots, o_{k,n}^{(t)}F^{(t)}M\right\}$. At time slot t, the EV n utilizes $E_{n,v}^{(t)}$ energy for executing EV computations, which can be given by:

$$E_{n,v}^{(t)} = \sum_{e=1}^{o_{k,n}^{(t)}F^{(t)}M} \chi f_{n,v}^2, \tag{8}$$

where the effective capacitance coefficient, denoted as χ, and the energy required for maintenance when the charging station k is in an idle state can be quantified as:

$$E_{k,i}^{(t)} = \sum_{e=1}^{o_{k,n}^{(t)}F^{(t)}M} \frac{p_k^m}{f_{n,v}}, \tag{9}$$

where p_k^m refers to the power required for maintenance when the charging station k is in an idle state. The total energy for charging station k for EV computation is expressed as:

$$E_{k,e}^{(t)} = E_{k,n}^{(t)} + E_{k,i}^{(t)}, \tag{10}$$

2.3. EV Charging Model

(1) *Computation Energy*: Based on the computation model outlined in Section 2.2, the charging station can be outfitted with a variety of energy harvesting technologies, including RF energy harvesters, photovoltaic modules, and wind turbines. These devices convert renewable resources such as ambient RF signals, wind, and solar energy into electrical power. To maintain equilibrium between power supply and demand, the station incorporates a battery. Moreover, the harvested renewable energy can be stored in this battery, supporting both local computing and computation offloading activities. The charging station k's total energy at time slot t is represented by $E_{k,c}^{(t)}$, where $E_{k,c}^{(t)} = \left(E_{k,l}^{(t)} + E_{k,e}^{(t)}\right)$.

(2) *Charging Energy*: To facilitate the guidance of the EV, a connection must be established between the charging station and the EV itself. $G_{n,k}^{(t)} \in \{0,1\}$ represents the association between the n-th EV and the k-th charging station at time slot t. When $G_{n,k}^{(t)} = 1$, it indicates a connection between the n-th EV and the k-th charging station, signifying that the EV is assigned to that station. Conversely, if $G_{n,k}^{(t)} = 0$, there is no correlation between the EV n and the charging station k. During the same time slot t, an EV must be exclusively allocated to one charging station, that is:

$$\sum_{k=1}^{K} G_{n,k}^{(t)} = 1. \tag{11}$$

Besides the energy requirements of the EV, the charging rate is another crucial factor, represented as follows:

$$v_k^{(t)} = \sum_{n=1}^{N} G_{n,k} v_n^{(t)}, \tag{12}$$

where the charging rate of EV n is denoted by $v_n^{(t)}$.

In an effort to mitigate the adverse effects of connecting charging stations to the power grid, renewable energy generators are installed in each station. Given the unpredictable and intermittent nature of renewable energy, this section details how each charging station is equipped with rechargeable batteries to harness and store this form of energy. The symbol $X_k^{(t)}$ denotes the production rate of renewable energy at charging station k, where $X_k^{(t)} \in [0, X_{k,\max}]$. We have:

$$X_k^{(t)} = X_{k,p}^{(t)} + X_{k,w}^{(t)}, \tag{13}$$

where $X_{k,p}^{(t)}$ represents the generation rate by the photovoltaic, which is given such that:

$$X_{kp}^{(t)} = A_k \cdot \delta^{(t)} \cdot e, \tag{14}$$

where A_k is the area of the solar panel, $\delta^{(t)}$ is the intensity of solar radiation at time slot t, and e is the efficiency of the solar panel. The power generated by the wind power is given such that:

$$X_{kw}^{(t)} = 0.5 \cdot \rho \cdot S_k \cdot {v^{(t)}}^3, \tag{15}$$

where ρ is the air density, S_k is the area covered by the blades of the wind turbine, and $v^{(t)}$ is the wind speed at time slot t.

Let $J_k^{(t)}$ represent the renewable energy stored in the battery of charging station k during time slot t, defined as $J_k^{(t)} \in \left[0, \min\left\{J_{k,\max}, \left(X_k^{(t)} - v_k^{(t)} - \frac{E_{k,c}^{(t)}}{\tau}\right)^+\right\}\right]$, where $(a)^+ \triangleq \max\{0, a\}$. If the rate of charging surpasses the generation rate of renewable energy,

the battery will not be able to store any energy. The energy equilibrium for the charging process of EVs is achieved, that is:

$$H_k^{(t)} = \left(v_k^{(t)} + \frac{E_{k,c}^{(t)}}{\tau} - X_k^{(t)} \right)^+ - l_k^{(t)}, \tag{16}$$

where $H_k^{(t)}$ is the energy purchased rate by the charging station k from the grid, τ is the time it takes for the computation energy of the charging station k at time slot t, and τ is usually equal to 1. $X_k^{(t)}$ denotes the renewable energy production rate at the k-th charging station, while $\left(v_k^{(t)} - X_k^{(t)}\right)^+$ represents the energy provided by the battery or the grid. Furthermore, $l_k^{(t)}$ denotes the energy conferred to the EV by the battery at the k-th charging station.

Let $L_k^{(t)} \in [0, L_{\max}]$ denote the present energy level of the battery at charging station k. $l_k^{(t)}$ represents the energy delivered to the EV by the k-th battery. Here, $l_k^{(t)}$ belongs to the interval $\left[0, L_k^{(t)}\right]$, and $L_k^{(t)}$ falls within the range of $(0, L_{k,\max}]$. The energy state of the battery for the k-th charging station $L_k^{(t)}$ should adhere to the equation as follows:

$$L_k^{(t+1)} = L_k^{(t)} - l_k^{(t)} + J_k^{(t)}, \tag{17}$$

where $l_k^{(t)}$ denotes the amount of energy withdrawn from the battery by charging station k; meanwhile, $J_k^{(t)}$ denotes the rate of renewable energy delivery to the battery. It is imperative to acknowledge the existence of an upper threshold pertaining to the amount of energy that can be extracted from a battery, that is:

$$0 \le l_k^{(t)} \le L_k^{(t)} \le L_{k,\max}. \tag{18}$$

The equation implies constraints on both the amount of energy the battery can supply and the battery's capacity.

To ensure the uninterrupted operation of charging stations, operators must devise suitable scheduling schemes to prevent a backlog in the charging demand of EVs. As EV requests occur randomly at any given moment, the relationship between charging requests and energy supply is illustrated by a virtual demand queue. $W_k^{(t)}$ represents the demand for charging station k during time slot t, which leads to the formation of the subsequent queue. The requirement for charging station k during time slot t is symbolized as $W_k^{(t)}$, subsequently resulting in the ensuing queue:

$$W_k^{(t+1)} = \max\left\{ W_k^{(t)} - v_k^{(t)}, 0 \right\} + \sum_{n=1}^{N} G_{n,k}^{(t)} h_n^{(t)}, \tag{19}$$

where $h_n^{(t)} \in [0, h_{\max}]$ represents the power requirement of the n-th EV. $\sum_{n=1}^{N} G_{n,k}^{(t)} h_n^{(t)}$ represents the charging requirement rate for accessing charging station k, while $v_k^{(t)}$ denotes the real-time charging rate at station k. The equilibrium of the queue is characterized as follows:

$$\bar{W}_k = \lim_{t \to \infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathrm{E}\{W_k^{(t)}\} < \infty. \tag{20}$$

If the queue is strongly stable, wherein $\mathrm{E}\{D_k^{(t)}\}$ denotes the anticipated value of queue $D_k^{(t)}$ [27], should the queue exhibit robust stability, it necessitates that the system is also

stable. Furthermore, a queue with robust stability must meet the equilibrium rate of the system's processing capability and stochastic arrival capacity, that is:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{t=0}^{t-1}\mathrm{E}\{\sum_{n=1}^{N}G_{n,k}^{(t)}h_n^{(t)}\}\leq\lim_{t\to\infty}\frac{1}{t}\sum_{t=0}^{t-1}\mathrm{E}\{v_k^{(t)}\}. \tag{21}$$

2.4. Problem Formulation

The primary objective of this study is to concurrently fine-tune the computation offloading decision, charging decision, and charging rate. The goal is to reduce the overall energy consumption as much as possible while ensuring that user latency requirements are met. We can derive the user's utility function. The utility of selecting EV n in time slot t is expressed as $U_n^{(t)}(G_{n,k},v_n)$. This utility function is influenced by factors such as the charging rate, task drop loss, energy consumption, and overall computation latency. It can be calculated as follows:

$$U_n^{(t)}(G_{n,k},v_n)=\ln\left[1+G_{n,k}^{(t)}v_n^{(t)}\right]-\beta\mathcal{T}_n^{(t)}-\eta E_{n,v}^{(t)}. \tag{22}$$

The weighting parameters for the computation latency and the energy for executing edge computations at the charging station k are represented by β and η, respectively. The total latency, denoted as $\mathcal{T}^{(t)}$, is quantified. This latency relies on two factors: the edge execution latency, symbolized as $T_{n,v}^{(t)}$, and the transmission delay to the chosen edge computing device, represented by $T_{k,n}^{(t)}$, that is, $\mathcal{T}_n^{(t)}=\left(T_{n,v}^{(t)}+T_{k,n}^{(t)}\right)$. There is also a certain constraint between the decision variables $G_{n,k}$ and $o_{k,n}$, i.e., $G_{n,k}$ is equal to zero, and $o_{k,n}$ must be equal to zero. This means that if EV n does not choose to charge at charging station k, then EV n will not perform the computation task, that is:

$$G_{n,k}\geq o_{k,n}. \tag{23}$$

The utility function for charging station k, representing the rate at which energy is purchased from the grid, can be defined as follows:

$$U_k^{(t)}(o_{k,n})=H_k^{(t)}. \tag{24}$$

The temporal coupling of battery charging and discharging is observed. This implies that the present state of battery charge–discharge will influence future conditions. This aspect amplifies the computational requirements of on-line scheduling, posing practical application challenges. Hence, in order to alleviate the stability constraint of $L_k(t)$, that is:

$$\lim_{t\to\infty}\frac{1}{t}\sum_{t=0}^{t-1}\mathrm{E}\left\{l_k^{(t)}\right\}\geq\lim_{t\to\infty}\frac{1}{t}\sum_{t=0}^{t-1}\mathrm{E}\left\{J_k^{(t)}\right\}. \tag{25}$$

The restrictions of queue $l_k^{(t)}$ are as per Equation (21), thereby liberating the battery's charging and discharging process from the constraints of previous time intervals.

Through optimizing the offloading factor $o_{k,n}^{(t)}$, the association between the n-th EV and the k-th CS $G_{n,k}^{(t)}$, and the charging rate of EV $v_n^{(t)}$, the utility function for EV n can be maximized, and the utility function for charging station k can be minimized. This optimization problem can be succinctly articulated as follows:

$$\max_{o_{k,n}^{(t)}, G_{n,k}^{(t)}, v_n^{(t)}} \mathbb{E}\left\{ \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \left[\sum_{n=1}^{N} \mu_n U_n^{(t)}(G_{n,k}, v_n) - \sum_{k=1}^{K} \mu_k U_k^{(t)}(o_{k,n}) \right] \right\}$$

$$\begin{aligned}
\text{s.t.} \quad & \text{C1:} \quad \sum_{k=1}^{K} G_{n,k}^{(t)} = 1, \\
& \text{C2:} \quad v_k^{(t)} = \sum_{n=1}^{N} G_{n,k} v_n^{(t)}, \\
& \text{C3:} \quad 0 \le l_k^{(t)} \le L_k^{(t)} \le L_{k,\max}, \\
& \text{C4:} \quad \bar{W}_k < \infty, \\
& \text{C5:} \quad \lim_{t\to\infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathrm{E}\left\{ \sum_{n=1}^{N} G_{n,k}^{(t)} h_n^{(t)} \right\} \le \lim_{t\to\infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathrm{E}\left\{ v_k^{(t)} \right\}, \\
& \text{C6:} \quad G_{n,k} \ge o_{k,n}, \\
& \text{C7:} \quad \lim_{t\to\infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathrm{E}\left\{ l_k^{(t)} \right\} \ge \lim_{t\to\infty} \frac{1}{t} \sum_{t=0}^{t-1} \mathrm{E}\left\{ J_k^{(t)} \right\},
\end{aligned} \tag{26}$$

where μ_k is the weight of the charging station k. $C1$ refers to the fact that an EV can select at most one charging station in a time slot t and only one charging station; $C2$—the charging rate to the k-th charging station is equal to the sum of the products of the rates of all EVs and the relationship factor with the k-th charging station; $C3$—the energy output from the charging station cannot exceed the total energy rates available at that time; $C4$, $C5$—the queue demand of charging station k needs to satisfy the strong stability condition; $C6$—there are constraints for the relationship factor and unloading factor in the unloading decision, where when the relationship factor is zero, EV computation must not be executed; $C7$—stability constraints for $L_k(t)$. The optimization problem involves both continuous variables (e.g., $v_n^{(t)}$) and discrete variables (e.g., $G_{n,k}^{(t)}$), making it a Mixed Integer Nonlinear Programming (MINLP) problem, which is inherently NP-hard.

3. MADDPG-Based EV Scheduling Algorithm

Traditional optimization algorithms, including linear optimization and convex optimization, have been widely applied to address resource allocation challenges in V2G systems. However, these algorithms are impractical for highly dynamic environments. This is because traditional algorithms have three main limitations compared to reinforcement learning algorithms: (1) Computationally difficult for real-time implementation. The computational complexity associated with traditional methods is higher due to the necessity of re-solving the problem each time. In contrast, reinforcement learning circumvents this issue by facilitating a streamlined process after training, where a mere input of moment-specific state parameters derives the desired output. (2) Reliance on accurate models and sensitive to parameter uncertainty. Traditional algorithms are not sensitive to the changes of some parameters in the environment, but reinforcement learning is different, as it can input the parameters of the environment as the state and can provide timely feedback to adapt to different states. (3) Inability to capture sequential decisions making in a mathematically exact way. To make a policy at every moment, the action is with time subscript t; therefore, the problem is a sequential decision. This kind of decision is difficult to solve with traditional optimization because the problem solved by traditional optimization can only be a problem in a certain time slot t. Solve a problem once in a certain time slot t, and then solve it again in the next time slot $t+1$. However, if a problem is a time decision, then it will have several time slots, and it is very difficult to call traditional optimization algorithms for each time slot.

To overcome the challenges posed by complex joint resource allocation problems, variants of reinforcement learning (RL) frameworks have emerged as highly effective solutions. These frameworks offer a promising approach to address the three main limitations

associated with such problems. The Deep Q-network (DQN) is a sophisticated RL algorithm designed for high-dimensional and discrete state spaces. However, it encounters difficulties when dealing with continuous action spaces because it requires quantization, which decreases precision and adds complexity. To address this issue, an Actor–Critic RL structure is employed. In this approach, the actor network generates actions using a deep neural network (DNN) at each time slot, while the critic network assesses the reward or Q-value of a specific state. The critic network aids the actor network by identifying preferable states, thereby guiding the agent towards more favorable outcomes. Within the Actor–Critic framework, three popular algorithms are Proximal Policy Optimization (PPO), Advantage Actor Critic (A2C), and Deep Deterministic Policy Gradient (DDPG). PPO is sample intensive, as it requires generating a new set of Q-values each time the policy is updated, rendering old Q-values obsolete. A2C employs the critic as a baseline using empirical Q-values to enhance training robustness by preventing back-propagation during random selection. In contrast, DDPG employs a deterministic policy where the gradients are calculated based on the Q-values provided by the critic network and the actions produced by the actor network.

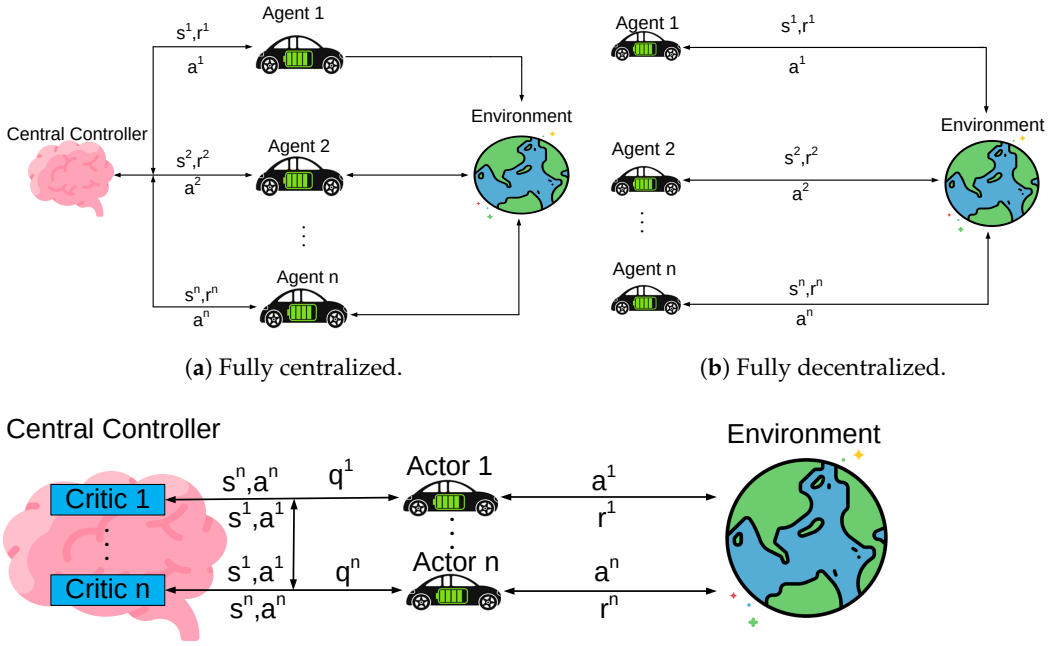
Additionally, DDPG is an off-policy algorithm, which enables it to leverage a large number of historical Q-values for training purposes. To incorporate RL-based methods for addressing (P0), we propose a reformulation of the problem within the RL framework. This framework consists of several critical components, including an agent, an environment, state space $\mathcal{S}$, action space $\mathcal{A}$, and reward function $\mathcal{R}$. The agent interacts with the environment by observing the state $\boldsymbol{s}(t) \in \mathcal{S}$ and selecting an action $\boldsymbol{a}(t) \in \mathcal{A}$ at each time slot t based on a policy $\mathbf{P}$, where $\mathbf{P} : \mathcal{S} \rightarrow \mathcal{A}$. As a result, the environment provides reward $\boldsymbol{r}(t)$ and transitions the state from $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$. In the MDP, the transition probability $\mathbb{P}(\boldsymbol{s}(t+1), \boldsymbol{r}(t) \mid \boldsymbol{s}(t), \boldsymbol{a}(t))$ denotes the chance of transitioning from state $\boldsymbol{s}(t)$ to $\boldsymbol{s}(t+1)$ and receiving reward $\boldsymbol{r}(t)$ after action $\boldsymbol{a}(t)$.

3.1. DEC-POMDP Framework

Figure 2 depicts various interactive relationships in multi-EV systems. In the fully centralized approach, action $\{a^1, a^2 \ldots a^n\}$ selection is overseen in a centralized manner, with the central controller maintaining a global Q-table. Conversely, the fully decentralized approach involves each EV independently learning its own policy using its individual Q-table. Unlike the previously mentioned methods, centralized training with decentralized execution employs a unique critic network for each agent. Each EV is an independent agent equipped with an actor network and a critic network and operates under the centralized training with decentralized execution (CTDE) mechanism. The exchange of critical state-action information is typically facilitated through shared communication channels or a central controller during training. Therefore, each EV agent collects information from all other EVs for centralized training, enabling each EV to evaluate actions using comprehensive global information. However, during the execution phase, each agent independently makes decisions based solely on its own actor network, ensuring decentralized operation. During training, a central controller gathers the states $\{s^1 \ldots s^n\}$, actions $\{a^1 \ldots a^n\}$, and rewards $\{r^1 \ldots r^n\}$ of all agents to assist in training the critic network. After training concludes, the central controller is no longer utilized, and each actor network independently makes decisions based on its own observations and associated critic network, without requiring communication with the central controller.

Figure 3 illustrates the reinforcement learning framework for resource allocation in computation offloading, where an electric vehicle (EV) acts as the agent interacting with the environment. To understand the MADDPG algorithm's training process, it is essential to first explore the design aspects of the centralized agent. Nevertheless, as the number of EVs and CSs increases, the state and action spaces will still expand exponentially, leading to inefficiency and potentially affecting latency and scalability. To address this, the problem is reformulated into a DEC-POMDP framework, with specific definitions

for the environmental state space, observation space, action space, and reward provided as follows:



(**a**) Fully centralized. (**b**) Fully decentralized.

(**c**) Centralized training with decentralized execution.

Figure 2. Interactive relationships in multi-EVs system.

- **Environmental State Space**: The state $\boldsymbol{s}^{(t)}$ can be precisely defined as

$$\boldsymbol{s}^{(t)} = \left\{ \boldsymbol{d}^{(t)}, \boldsymbol{g}^{(t)}, \boldsymbol{F}^{(t)}, \delta^{(t)}, v^{(t)}, \boldsymbol{h}^{(t)} \right\}, \tag{27}$$

where $\boldsymbol{d}^{(t)}$ represents the Euclidean distances between all EVs and all CSs; similarly, $\boldsymbol{g}^{(t)}$ represents all the channel gains. $\boldsymbol{F}^{(t)} = \{F_1^{(t)}, \ldots, F_k^{(t)}, \ldots, F_K^{(t)}\}$ represents the computation tasks of all charging stations, correlated with a data volume; $\delta^{(t)}$ and $v^{(t)}$ correspond to the intensity of solar radiation and the wind speed at time slot t, respectively; and $\boldsymbol{h}^{(t)} = \{h_1^{(t)}, \ldots, h_n^{(t)} \ldots, h_N^{(t)}\}$ represents the power demand of all EVs.

- **Observation Space**: The state $\boldsymbol{s}_n^{(t)}$ can be precisely defined as

$$\boldsymbol{s}_n^{(t)} = \left\{ \boldsymbol{d}_n^{(t)}, \boldsymbol{g}_n^{(t)}, \boldsymbol{F}^{(t)}, \delta^{(t)}, v^{(t)}, h_n^{(t)} \right\}, \tag{28}$$

where $\boldsymbol{d}_n^{(t)} = \{d_{n,1}^{(t)}, \ldots, d_{n,k}^{(t)}, \ldots, d_{n,K}^{(t)}\}$ represents the Euclidean distances from EV n to all CSs. Similarly, $\boldsymbol{g}_n^{(t)} = \{g_{n,1}^{(t)}, \ldots, g_{n,k}^{(t)}, \ldots, g_{n,K}^{(t)}\}$ represents the channel gains from EV n to all CSs. $\boldsymbol{F}^{(t)} = \{F_1^{(t)}, \ldots, F_k^{(t)}, \ldots, F_K^{(t)}\}$ represents the computation tasks of all charging stations, correlated with a data volume; $\delta^{(t)}$ and $v^{(t)}$ correspond to the intensity of solar radiation and the wind speed at time slot t, respectively; and $h_n^{(t)}$ represents the power demand of the n-th EV.

- **Action space**: As the agent, the EV makes decisions regarding its actions at every time slot. These decisions encompass whether to select a charging station for charging, as well as whether to compute the unloading task and charging rate subsequent to

selecting a charging station for charging. Therefore, the action performed by each individual EV n can be precisely defined as:

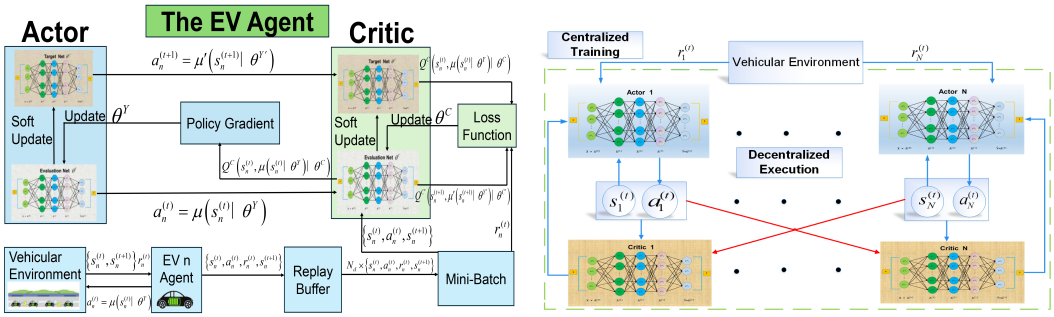
$$\boldsymbol{a}_n^{(t)} = \left\{ o_{k,n}^{(t)}, G_{n,k}^{(t)}, v_n^{(t)} \right\}. \tag{29}$$

Based on the given statement, we observe that the mentioned action involves several variables. Firstly, there is a discrete binary offloading factor variable, denoted as $o_k^{(t)}$, then the association variable $G_{k,n}^{(t)}$ between the n-th EV and the k-th charging station during a specific time slot t. Additionally, we have the continuous charging rate of EV n, represented as variable $v_n^{(t)}$.

- **Reward function**: To maximize the system's overall utility, the reward function is designed to align with this objective and can be formulated as follows during time slot t:

$$\boldsymbol{r}_n^{(t)} = \begin{cases} U_n^t - \mu_k \sum_{k \in \{1,2,\ldots,K\}} U_k^t, & \text{if all the constraints} \\ & \text{are satisfied,} \\ C^{(t)}, & \text{otherwise.} \end{cases} \tag{30}$$

where $C^{(t)}$ is a negative constant at each time slot t. μ_k is the weight of the charging station k. In the optimization problem, constraints C1, C2, and C6 can be satisfied through the design of the action space. For the remaining constraints, C3, C4, C5, and C7, we determine the reward function's output based on whether all these constraints are satisfied. If all are met, a positive value is returned; if any one is not satisfied, a negative value is returned. Specifically, for constraints C5 and C7, we save and accumulate the relevant data from previous time steps, such as $\mathrm{E}\left\{ v_k^{(t)} \right\}$ and $\mathrm{E}\left\{ l_k^{(t)} \right\}$, when calculating the reward at each time step t. When the time step t tends to infinity, we set a very large number, 10,000, to simulate infinity.



(**a**) (**b**)

Figure 3. Framework of the MADDPG. (**a**) Framework of the MADDPG algorithm. (**b**) Neural network architecture.

3.2. Preliminary of DDPG Algorithm

The EV agent uses the DDPG algorithm, combining the strengths of policy gradient and DQN. Unlike the DQN Q network, DDPG uses two frameworks: the actor network for decisions and the critic network for evaluating behavior. Both networks include evaluation networks Y and C and target networks Y' and C', with parameters θ^Y, θ^C, $\theta^{Y'}$, and $\theta^{C'}$. Updating parameters slower than the online network, the fixed target network stabilizes training. Random sampling from the experience replay buffer reduces sample correlation. As a deterministic algorithm, DDPG outputs unique actions for the same state, enhancing efficiency and reducing sample needs. This approach can extend to other EV agents using DDPG.

The critic network employed in DDPG employs a methodology known as value function evaluation, which bears resemblance to the evaluation network found in DQN.

The action selected by the actor is then evaluated by the critic, another essential component, using the state-action function $Q^C(\cdot)$: the input state to the EV, denoted as $\boldsymbol{s}_n^{(t)}$, and the immediate reward $\boldsymbol{r}_n^{(t)}$ to which the discount factor γ is applied. A deep neural network (DNN) is utilized by the critic network to estimate the action-state-value function $Q^C(s,a) = \mathbb{E}\{R(s,a)\}$, where $R(s,a) = \sum_{t=0}^{\infty} \gamma^t \boldsymbol{r}_n^{(t)}$ represents the accumulated reward. This reward, computed recursively using the Bellman equation, is defined as:

$$\mathcal{Q}^C(s,a) = \mathbb{E}\Big\{r_n^{(t)}\Big(s_n^{(t)}, a_n^{(t)}\Big) + \gamma \mathbb{E}\Big\{\mathcal{Q}^C\Big(s_n^{(t+1)}, a_n^{(t+1)}\Big)\Big\}\Big\}. \tag{31}$$

Target networks and experience replay enhance the stability of DDPG. As shown in Figure 2, both the actor and the critic use two deep neural networks (DNNs), specifically, an evaluation network and a target network. An experience replay buffer R with capacity N_d stores transitions. DDPG aims to determine the optimal policy π_n^* and learn the corresponding state-action function by iteratively adjusting the parameters of the actor's and critic's evaluation and target networks until convergence is achieved. The evaluation networks' parameters, θ^Y and θ^C, are updated in real time. The agent is fed with a random mini-batch of transitions, of fixed size N_m, drawn from the replay buffer. During training, the evaluation networks' parameters are updated by the actor and critic based on each transition.

At time slot t, taking the t-th transition, $\{s_n^{(t)}, a_n^{(t)}, r_n^{(t)}, s_n^{(t+1)}\}$, as an example, the critic updates the parameters of the evaluation network by minimizing the loss:

$$L(\theta^C) = \mathbb{E}\big[(Q^C(s_n^{(t)}, a_n^{(t)} \mid \theta^C) - y_x)^2\big] \tag{32}$$

where y_x can be computed as:

$$y_x = r_n^{(t)}\big(s_n^{(t)}, a_n^{(t)}\big) + \gamma Q^{C'}\big(s_n^{(t+1)}, \mu'(s_n^{(t+1)} \mid \theta^{Y'}) \mid \theta^{C'}\big). \tag{33}$$

where $Q^{C'}(\cdot)$ represents the state-action function for the target network. The target network, carrying the same structure as the evaluation network, is utilized to compute the updated objectives. It is crucial to note that this value-based DRL approach is unsuitable for optimizing continuous variables, and therefore cannot be directly applied to the joint optimization problem we have outlined. Specifically, if the loss function $L(\theta^C)$ is continuously differentiable, the parameter θ^C can be modified using its gradient [28]. In DDPG, the actor network employs a policy search-based method to select an optimal action, following the deterministic policy $an^{(t)} = \mu\big(s_n{}^{(t)} \mid \theta^Y\big)$ given the current state $s_n^{(t)}$. The primary concept of the DDPG actor network is to update the policy parameters θ^Y by maximizing the policy objective function:

$$J(\boldsymbol{\theta}^Y) = \mathbb{E}[Q^C\big(s_n^{(t)}, a_n^{(t)} \mid \theta^C\big) \mid a_n^{(t)} = \mu\big(s_n^{(t)} \mid \theta^Y\big)], \tag{34}$$

By implementing steps toward the direction of $\nabla_{\boldsymbol{\theta}^Y} J\big(\boldsymbol{\theta}^Y\big)$, that is:

$$\nabla_{\theta^Y} J \approx \mathbb{E}[\nabla_a \mathcal{Q}(s_n, a_n = \mu(s_n|\theta^Y)|\theta^C) \cdot \nabla_{\theta^Y} \mu(s_n|\theta^Y)] \tag{35}$$

The gradient comprises two components: (1) the gradient inherent within the critic's action value function $\big(\nabla_a \mathcal{Q}\big(s_n^{(t)}, a_n^{(t)} = \mu\big(s_n^{(t)} \mid \theta^Y\big) \mid \theta^C\big)$, and (2) the gradient associated with its deterministic strategy action $\big(\nabla_{\theta^Y} \mu\big(s_n^{(t)} \mid \theta^Y\big)\big)$. By leveraging the real-time updated parameters θ^Y and θ^C of the target networks, namely $\theta^{Y'}$ and $\theta^{C'}$, a soft update can be performed in the following manner:

$$\begin{aligned} \theta^{Y'} &\leftarrow \tau_1 \theta^Y + (1-\tau_1)\theta^{Y'} \\ \theta^{C'} &\leftarrow \tau_2 \theta^C + (1-\tau_2)\theta^{C'}. \end{aligned} \tag{36}$$

where the stability of learning can be improved by introducing a soft update factor, denoted as τ_1, τ_2, in the equation, and the constant τ_1 and τ_2 close to 0.

3.3. MADDPG Framework for EVs

Based on the analysis conducted above, it appears feasible to directly apply single DDPG to address our multi-agent optimization problem. This approach involves allowing each agent to independently learn its own Q-value function. However, directly applying DDPG to each agent can create a fully distributed scenario where agents make decisions in isolation, ignoring the influence of others. As a result, this can lead to local optima, and the environment may appear nonstationary to each individual agent. To address these issues, we utilize MADDPG, which learns a Q-value function for each agent by incorporating global information. This method allows us to effectively tackle our joint optimization problem within a multi-agent framework. In the case of MADDPG, the training stage involves possessing complete knowledge of the actions and states of all agents. As a result, the environment is considered stationary, even if there are changes in the policy. This means that the learning process remains consistent and unaffected by shifting policies, which mean:

$$\begin{aligned}&\mathcal{P}(\mathbf{s_n^{t+1}} \mid \mathbf{s_n^t}, \mathbf{a_1}^{(t)}, \mathbf{a_2}^{(t)}, \ldots, \mathbf{a_N}^{(t)}, \pi_1^*, \pi_2^*, \ldots, \pi_N^*)\\ &= \mathcal{P}(\mathbf{s_n^{t+1}} \mid \mathbf{s_n^t}, \mathbf{a_1}^{(t)}, \mathbf{a_2}^{(t)}, \ldots, \mathbf{a_N}^{(t)})\\ &= \mathcal{P}(\mathbf{s_n^{t+1}} \mid \mathbf{s_n^t}, \mathbf{a_1}^{(t)}, \mathbf{a_2}^{(t)}, \ldots, \mathbf{a_N}^{(t)}, \dot{\pi_1}^*, \dot{\pi_2}^*, \ldots, \dot{\pi_N}^*)\end{aligned} \tag{37}$$

where $\pi_i^* \neq \dot{\pi_i^*}$. This section provides a comprehensive description of the MADDPG algorithm, including its intricate details and workings.

As illustrated in Figure 3, the MADDPG framework comprises N agents and an environment. Each agent undergoes two distinct phases: (1) the centralized training phase, and (2) the decentralized execution phase. It is essential to note that the training phase is conducted offline, during which exploration is necessary to discover the optimal policy. During the execution phase, the MADDPG algorithm solely utilizes forward propagation and does not involve a random exploration process. This execution phase requires significantly fewer resources compared to the training phase. Following this, we will use the example of an agent to explain the method of centrally training the MADDPG model and then deploying the acquired model in a decentralized manner.

In the centralized approach during offline training, the critic network calculates the centralized action-value function $Q(\mathbf{S}, \mathbf{A} \mid \boldsymbol{\theta^C})$. This computation is derived from integral state data such as the actions and states of every single agent. The centralized Q function appraises the actions taken by the actor on a broad scale, using this information to instruct the actor to make superior decisions. Subsequently, the critic network adjusts the parameters $\boldsymbol{\theta^C}$ through the process of minimizing the associated loss:

$$L(\theta^C) = \mathbb{E}[(Q^C(\mathbf{S^{(t)}}, \mathbf{A^{(t)}} \mid \boldsymbol{\theta^C}) - y_x^M)^2] \tag{38}$$

and

$$y_x^M = r^{(t)}(\mathbf{S^{(t)}}, \mathbf{A^{(t)}}) + \gamma Q^{C'}(\mathbf{S^{(t+1)}}, \mu'(\mathbf{S^{(t+1)}} \mid \boldsymbol{\theta^{Y'}}) \mid \boldsymbol{\theta^{C'}}). \tag{39}$$

Here, $\mathbf{S^{(t)}} = (\mathbf{s_1^t}, \mathbf{s_2^t}, \ldots, \mathbf{s_N^t})$ represents the current states, $\mathbf{A^{(t)}} = (\mathbf{a_1^t}, \mathbf{a_2^t}, \ldots, \mathbf{a_N^t})$ represents the current action, and $\boldsymbol{\theta^C}$ denotes the parameter of the evaluation network. Additionally, $\mathbf{S^{(t+1)}} = (\mathbf{s_1^t}, \mathbf{s_2^t}, \ldots, \mathbf{s_N^t})$ refers to the updated states for the target network, and $\boldsymbol{\theta^{C'}}$ represents the parameter of the target network.

Simultaneously, the actor network refines its parameters $\boldsymbol{\theta^Y}$ and generates actions $\mathbf{A^{(t)}}$ using the centralized Q function obtained from the critic's evaluation, along with its own observation data. More specifically, the actor network effectively modifies the network parameters $\boldsymbol{\theta^Y}$ by following the direction provided by $\nabla_{\boldsymbol{\theta^Y}} J(\boldsymbol{\theta^Y})$ as defined by

$$\nabla_{\boldsymbol{\theta^Y}} J(\boldsymbol{\theta^Y}) \approx \mathbb{E}\left[\nabla_a Q(\mathbf{S^{(t)}}, \mathbf{A^{(t)}} = \mu(\mathbf{S^{(t)}} \mid \boldsymbol{\theta^Y}) \mid \boldsymbol{\theta^C}) \times \nabla_{\boldsymbol{\theta^Y}} \mu(\mathbf{S^{(t)}} \mid \boldsymbol{\theta^Y})\right], \tag{40}$$

and

$$J(\boldsymbol{\theta}^{Y}) = \mathbb{E}[Q^{C}(\mathbf{S}^{(\mathbf{t})}, \mathbf{A}^{(\mathbf{t})} \mid \boldsymbol{\theta}^{C}) \mid \mathbf{A}^{(\mathbf{t})} = \mu(\mathbf{S}^{(\mathbf{t})} \mid \boldsymbol{\theta}^{Y})] \tag{41}$$

During the decentralized execution phase, the critic network is not utilized, and only the trained actor network operates in real-time. The actor network generates actions based on its current state. During execution, only a forward propagation process is involved, without any random exploration process. This approach significantly lowers the consumption of computing resources and time as compared to the training phase. With finely tuned parameters, each agent can determine an action that closely approximates the global optimum, without needing information about other agents. The entire algorithm is detailed in Algorithm 1.

Algorithm 1 Multi-Agent Deep Deterministic Policy Gradient (MADDPG).

/* *Initialization* */
Initialize Number of EV agents N, discount factor γ, soft update parameter ϵ_1, ϵ_2
Initialize the actor evaluation networks θ^{Y} and critic evaluation networks θ^{C} for each agent n
Initialize target actor networks $\theta^{Y'}$ and target critic networks $\theta^{C'}$ for each agent n
Initialize replay buffer R with fixed size N_d
/* *Model Training* */
for each episode **do**
  Receive initial observations $s_n^{(t)}$
  for each timestep **do**
    each agent n, select action $a_n^{(t)} = \mu\left(s_n^{(t)} \mid \theta^{Y}\right)$ and execute it
    Observe joint reward $r_n^{(t)}$ and new state $s_n^{(t+1)}$
    Store tuple $\left\{s_n^{(t)}, a_n^{(t)}, r_n^{(t)}, s_n^{(t+1)}\right\}$ in replay buffer R. Set $s_n^{(t)} \leftarrow s_n^{(t+1)}$.
    for each agent n **do**
      if the number of tuples $> N_d$ **then**
        Sample a mini-batch of N_m tuples from R
        Updates the parameters of the evaluation network by minimizing the loss:

$$L(\theta^{C}) = \mathbb{E}[(Q^{C}(\mathbf{S}^{(\mathbf{t})}, \mathbf{A}^{(\mathbf{t})} \mid \boldsymbol{\theta}^{C}) - y_x^{M})^2]$$

        and:

$$y_x^{M} = r^{(t)}(\mathbf{S}^{(\mathbf{t})}, \mathbf{A}^{(\mathbf{t})}) + \gamma Q^{C'}(\mathbf{S}^{(\mathbf{t+1})}, \mu'(\mathbf{S}^{(\mathbf{t+1})} \mid \boldsymbol{\theta}^{Y'}) \mid \boldsymbol{\theta}^{C'})$$

        Update the actor network by using the policy gradient:

$$\nabla_{\boldsymbol{\theta}^{Y}} J(\boldsymbol{\theta}^{Y}) \approx \mathbb{E}\left[\nabla_a Q(\mathbf{S}^{(\mathbf{t})}, \mathbf{A}^{(\mathbf{t})} = \mu(\mathbf{S}^{(\mathbf{t})} \mid \boldsymbol{\theta}^{Y}) \mid \boldsymbol{\theta}^{C}) \times \nabla_{\boldsymbol{\theta}^{Y}} \mu(\mathbf{S}^{(\mathbf{t})} \mid \boldsymbol{\theta}^{Y})\right]$$

        Update actor's and critic's target networks parameters according to Equation (32).
      end if
    end for
  end for
end for

4. Experimental Results

This section details the framework of the MADDPG algorithm, demonstrated through numerical simulations under the proposed centralized training with the decentralized execution (CTDE) mechanism within a vehicular network system supporting EVs. Initially, we introduce the setup of the simulation parameters. Subsequently, we validate the performance of the MADDPG algorithm in various scenarios and compare it with other benchmark schemes. The software environment was set up with Python version 3.9.13 and TensorFlow version 2.15.0.

4.1. Simulation Setup

For the simulations, the hardware environment comprised a server equipped with a 12th Gen Intel(R) Core(TM) i9-12900H processor operating at 2.50 GHz, and a memory configuration of 32.0 GB RAM, which was manufactured in China by XBDCT (Shenzhen) Technology Co., Ltd. (Shenzhen, China).

In the vehicular network system supporting EVs, we consider a square area where $N = 8$ electric vehicles are randomly distributed within a [100, 100] region [29]. The total duration $T = 320$ s is divided into $s = 40$ time slots. The channel gain is set to $\phi_1 = -50$ dB with a reference distance of 1 m [27]. The transmission bandwidth is configured at $B = 5$ MHz. The noise power is assumed to be $\sigma^2 = -100$ dBm. We set the transmission power of charging stations to $P_{\text{down}} = 0.5$ W and specify the required CPU cycles per bit at $s = 1000$ cycles/bit [30].

Additionally, we configure the computing capabilities of the charging station and the MEC server with $f_{k,e} = 0.6$ GHz and $f_{n,v} = 1.2$ GHz [30], respectively. To implement the MADDPG algorithm, the architectures of the policy and critic networks are detailed as follows. Each agent's actor network and critic network include three fully connected hidden layers, consisting of 256, 64, and 16 neurons, respectively. The activation function utilized is the Rectified Linear Unit (ReLU), and the network weights are updated using the Adam optimizer. The algorithm was trained across a total of 1000 episodes. For the purpose of evaluating performance, the following three benchmarks are implemented. The detailed parameter design is presented in Table 2.

- DDPG under fully centralized mechanism **(FC-DDPG)**: The system jointly determines the task offloading decisions and computation resource allocations by employing a DDPG agent. This agent processes global system states as inputs, with the utility function acting as the reward mechanism.
- DDPG under fully decentralized mechanism **(FD-DDPG)**: The MADDPG algorithm, under a purely decentralized mechanism, allows each electric vehicle (EV) to act as an intelligent agent. These EV agents independently determine task offloading decisions and computation resource allocations. The system implements DDPG within each EV, using the utility function as the reward metric for these agents. This approach enables decentralized decision-making, critical for scalable and efficient resource management in vehicular networks.
- Actor–Critic-based algorithm **(Actor–Critic)**: To evaluate the performance of the proposed MADDPG under the CTDE mechanism for computation offloading, we implement the Actor–Critic-based algorithm. This continuous action space RL algorithm addresses the computation offloading problem, enabling robust comparisons of continuous action dynamics.

Table 2. Simulation parameters.

System Model Parameters	
Parameter	**Value**
Number of electric vehicles N	8
Number of charging stations K	4
Time period T	320 s
Number of time slots s	40
Task deadline t_k	[5, 10] s
EV speed V	10 m/s
Transmission bandwidth B	5 MHz
Channel gain ϕ_1	-50 dB
Noise power σ^2	-100 dBm
Transmission power P_{down}	0.5 W
CPU cycles per bit s	1000 cycles/bit
Computational capability of charging stations $f_{k,e}$	0.6 GHz
Computational capability of electric vehicles $f_{n,v}$	1.2 GHz

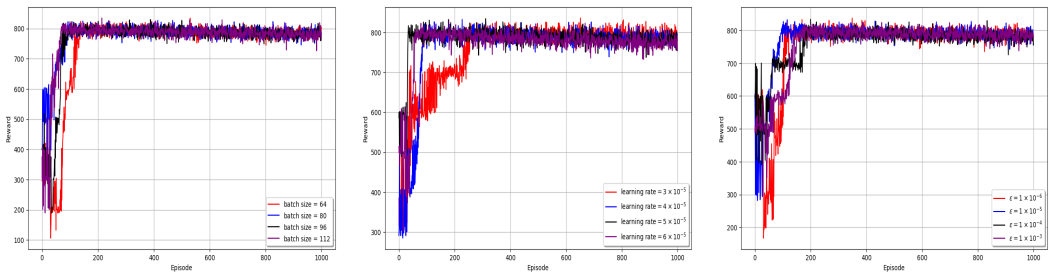
Table 2. *Cont.*

MADDPG Parameters	
Parameter	**Value**
Discount factor γ	0.999
Batch size N_m	112
Replay buffer size N_d	1×10^4
Actor network learning rate	5×10^{-5}
Critic network learning rate	1×10^{-4}
Exploration constant ϵ	1×10^{-5}
Soft update factor τ	3×10^{-4}

4.2. Convergence Performance

Figure 4 illustrates the convergence performance of the MADDPG algorithm under different parameters (batch size, exploration rate, and learning rate). In Figure 4a, smaller batch sizes 64 result in slower convergence and more fluctuations, whereas larger batch sizes 96 and 112 lead to faster and more stable convergence. Batch size 80 shows intermediate performance. This is because smaller batch sizes cause higher variance in gradient estimation, affecting stability and speed, while larger batch sizes provide more stable gradient estimates, accelerating convergence but increasing computational overhead. In Figure 4b, a learning rate of 5×10^{-5} yields the fastest convergence and highest rewards. Lower rates (3×10^{-5} and 4×10^{-5}) slow down convergence, while a higher rate (6×10^{-5}) causes instability and fluctuations. Proper learning rates maintain stable updates and accelerate convergence, while too low rates slow down learning, and too high rates lead to gradient explosion or oscillations, affecting stability. In Figure 4c, with an exploration rate of 1×10^{-4}, the convergence speed is fastest and the final reward is highest. Lower exploration rates (1×10^{-5} and 1×10^{-6}) result in slower convergence, while a higher rate (1×10^{-3}) causes significant fluctuations. Appropriate exploration rates balance exploration and exploitation, steadily increasing rewards, whereas too low rates lead to premature convergence to suboptimal solutions, and too high rates result in excessive exploration, destabilizing the process.

Figure 5 illustrates the convergence performance of the MADDPG algorithm by comparing the reward per episode for three schemes: FC-DDPG scheme, FD-DDPG scheme, and Actor–Critic scheme. The MADDPG scheme demonstrates the highest and most stable rewards, indicating superior convergence behavior compared to the other schemes. The FD-DDPG scheme shows rapid initial learning but stabilizes at a lower reward level than the MADDPG scheme, suggesting that full decentralization might limit the ability to optimize the global policy effectively. The FC-DDPG scheme exhibits significant fluctuations in rewards, indicating instability and difficulty in converging to a consistent policy. The Actor–Critic scheme, while more stable than FC-DDPG, also stabilizes at a lower reward level compared to the MADDPG scheme. These results highlight the advantage of the MADDPG approach in balancing centralized training and decentralized execution, leading to better overall performance.



(**a**) Batch size. (**b**) Learning rate. (**c**) Exploration constant.

Figure 4. Convergence performance with different parameters.

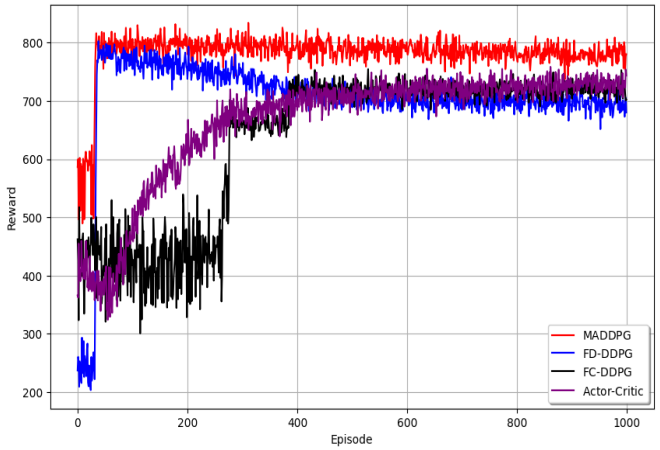
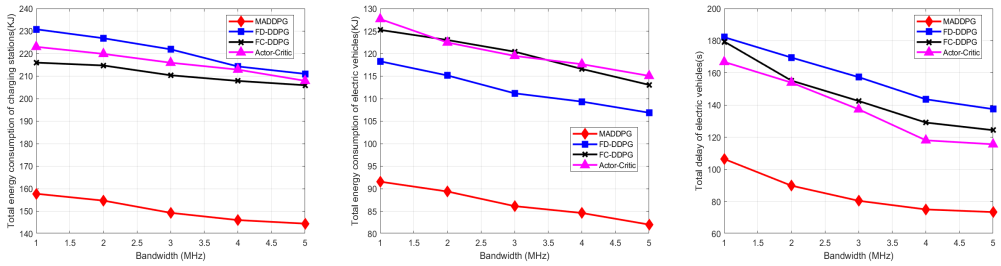


Figure 5. Convergence performance of MADDPG with benchmarks.

4.3. System Performance Comparison

In Figure 6, we evaluate the impact of varying bandwidth on the performance of MADDPG, FD-DDPG, FC-DDPG, and Actor–Critic in EVs assisting CSs with computation offloading. The analysis shows that as bandwidth increases, energy consumption for both CSs and EVs decreases, and EV delays are reduced. These improvements are primarily due to enhanced data transmission efficiency facilitated by greater bandwidth. Beyond 4 MHz, the improvements plateau, indicating bandwidth saturation, where further increases offer minimal gains. This plateau suggests that system performance becomes constrained by other factors, such as the limitations of algorithmic management or the physical capabilities of the infrastructure. MADDPG surpasses other methods due to its hybrid structure that integrates centralized training with distributed execution. This design enables the optimization of resource allocation strategies by learning and coordinating the policies of other users during the training phase. FD-DDPG struggles at low bandwidths due to its lack of centralized control. FC-DDPG excels in reducing CS energy consumption but is less effective for EVs due to its conservative resource scheduling. Actor–Critic is balanced but less effective in complex scenarios, likely due to slower policy updates. This analysis highlights the importance of bandwidth in computation offloading efficiency and the need to choose algorithms that match specific performance goals and infrastructure capabilities.

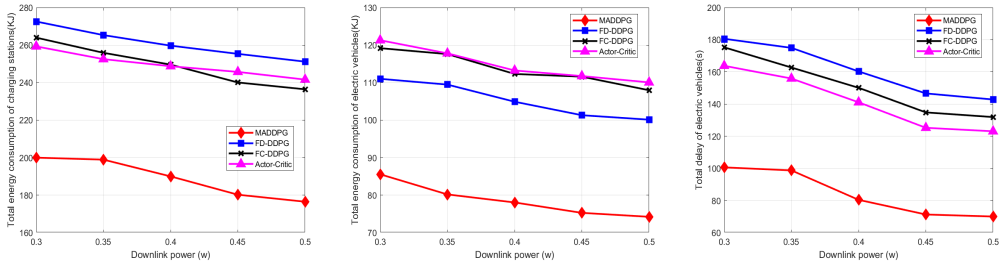


(**a**) CS energy consumption. (**b**) EV energy consumption. (**c**) System delay.

Figure 6. Performance comparison under different bandwidth.

Figure 7 shows that higher downlink power reduces energy consumption and delays for both charging stations and electric vehicles due to improved data transmission efficiency. For example, increasing downlink power from 0.3 *W* to 0.5 *W* boosts transmission rates, reducing energy consumption during local task processing. MADDPG consistently surpasses other algorithms in terms of reducing energy consumption and delay, enabling each agent to learn the policies of other agents throughout the training stage. This approach partially mitigates the issues arising from the interdependent nature of optimization problems. FD-DDPG and FC-DDPG are less effective in resource coordination and responsiveness.

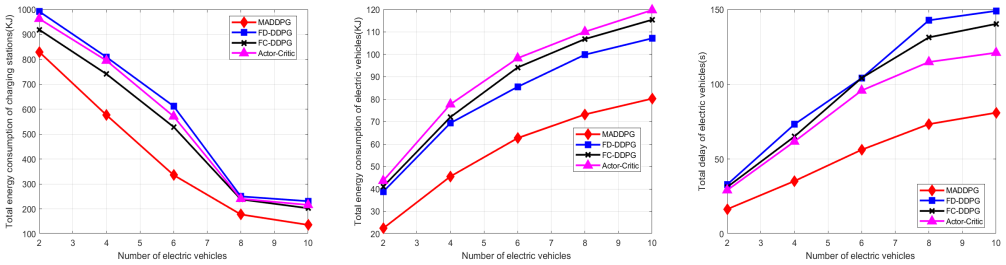
FD-DDPG has low resource utilization at lower power levels, while the centralized decision-making of FC-DDPG is slower in dynamic settings. Actor–Critic reduces delays under certain conditions but is slower in complex scenarios. Beyond 0.45 W, delay reduction plateaus, indicating a saturation point in power utilization efficiency.



(**a**) CS energy consumption. (**b**) EV energy consumption. (**c**) System delay.

Figure 7. Performance comparison under different downlink power.

Figure 8 illustrates that as the number of electric vehicles increases, the charging stations' total energy consumption initially decreases significantly due to distributed processing, reducing their load and energy demand. However, beyond eight vehicles, this reduction levels off, indicating a saturation point, beyond which additional vehicles have minimal impact on energy savings. Conversely, energy consumption and delay for EVs increase with more vehicles due to additional processing energy and higher computation and transmission delays. The MADDPG algorithm outperforms others across all metrics by allowing each agent's critic to evaluate the joint policy's impact on expected rewards. Ensemble training enhances generalization with unknown agents. In contrast, the purely distributed decision-making of FD-DDPG lacks centralized coordination, resulting in inefficient resource allocation and limited energy and delay reductions. FC-DDPG addresses nonstationarity but faces high communication overhead and potential data staleness. Actor–Critic performs well initially but struggles to adapt quickly in complex scenarios, especially as the number of vehicles increases.



(**a**) CS energy consumption. (**b**) EV energy consumption. (**c**) System delay.

Figure 8. Performance comparison under different number of EVs.

5. Conclusions

In this paper, we examined the joint optimization of EV charging scheduling and computation offloading at charging stations. We established a system model with offloading decisions, charging rates, and charging decisions as EV optimization decisions with practical constraints. We designed utility functions centered on energy consumption and delay, converting the issue into a mixed-integer nonlinear program (MINLP). Traditional optimization techniques struggle with the complexity of MINLP, so we used a reinforcement learning (RL) approach, which handles environmental uncertainties without relying on the system model. To address slow RL convergence, we integrated deep learning with the Actor–Critic technique, improving convergence by converting value function table maintenance into neural network training. We implemented four DRL algorithms, MADDPG, FD-DDPG, FC-DDPG, and traditional Actor–Critic, and conducted performance

comparisons. Simulation results show that MADDPG rapidly converges and outperforms benchmarks in minimizing system energy consumption and delay. Our research offers an effective solution for optimizing EV charging scheduling and computation offloading, demonstrating high practical value and potential for widespread adoption.

For future work, we plan to compare the proposed MADDPG-based approach with other multi-agent DRL algorithms to assess its performance across diverse scenarios. Additionally, enhancing the algorithm's adaptability to more dynamic vehicular environments, including varying vehicle densities and network topologies, will be crucial. Further exploration of objectives like fairness in resource distribution and scalability could also provide deeper insights, making the approach more applicable to real-world VEC networks.

Author Contributions: Conceptualization, Z.Z. and C.Y.; methodology, Z.Z. and C.Y.; software, Z.Z. and B.T.; validation, Z.Z. and B.T.; formal analysis, Z.Z. and C.Y.; writing—original draft preparation, Z.Z. and C.Y.; writing—review and editing, Z.Z.; funding acquisition, C.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research and Innovation Team of Chongqing University of Technology (2023TDZ003), the Chongqing Natural Science Foundation Innovation Development Joint Fund (CSTB2023NSCQ-LMX0014), and the High-End Foreign Experts Project (No. GDW20165200063).

Data Availability Statement: Data are contained within the article.

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

References

1. Satyanarayanan, M. The emergence of edge computing. *Computer* **2017**, *50*, 30–39. [CrossRef]
2. Saraswat, S.; Gupta, H.P.; Dutta, T.; Das, S.K. Energy efficient data forwarding scheme in fog-based ubiquitous system with deadline constraints. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 213–226. [CrossRef]
3. Zhou, H.; Jiang, K.; Liu, X.; Li, X.; Leung, V. Deep reinforcement learning for energy-efficient computation offloading in mobile-edge computing. *IEEE Internet Things J.* **2022**, *9*, 1517–1530. [CrossRef]
4. Ali, H.S.; Rout, R.R.; Parimi, P.; Das, S.K. Real-time task scheduling in fog-cloud computing framework for IoT applications: A fuzzy logic based approach. In Proceedings of the International Conference on Communication Systems and Networks (COMSNETS), Bangalore, India, 5–9 January 2021; pp. 556–564.
5. Ahani, G.; Yuan, D. BS-assisted task offloading for D2D networks with presence of user mobility. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; pp. 1–5.
6. Dinh, T.Q.; Tang, J.; La, Q.D.; Quek, T.Q.S. Offloading in mobile edge computing: Task allocation and computational frequency scaling. *IEEE Trans. Commun.* **2017**, *65*, 3571–3584.
7. Yan, J.; Bi, S.; Zhang, Y.J.; Tao, M. Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 235–250. [CrossRef]
8. Lyu, X.; Ren, C.; Ni, W.; Tian, H.; Liu, R.P.; Dutkiewicz, E. Optimal online data partitioning for geo-distributed machine learning in edge of wireless networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2393–2406. [CrossRef]
9. Ale, L.; Zhang, N.; Wu, H.; Chen, D.; Han, T. Online proactive caching in mobile edge computing using bidirectional deep recurrent neural network. *IEEE Internet Things J.* **2019**, *6*, 5520–5530. [CrossRef]
10. Zhou, B.; Zou, J.; Chung, C.Y.; Wang, H.; Liu, N.; Voropai, N.; Xu, D. Multi-microgrid energy management systems: Architecture, communication, and scheduling strategies. *J. Mod. Power Syst. Clean Energy* **2021**, *9*, 463–476. [CrossRef]
11. Sun, B.; Huang, Z.; Tan, X.; Tsang, D.H.K. Optimal scheduling for electric vehicle charging with discrete charging levels in distribution grid. *IEEE Trans. Smart Grid* **2018**, *9*, 624–634. [CrossRef]
12. Hou, X.; Wang, J.; Huang, T.; Wang, T.; Wang, P. Smart home energy management optimization method considering energy storage and electric vehicle. *IEEE Access* **2019**, *7*, 144010–144020. [CrossRef]
13. Kasani, V.S.; Tiwari, D.; Khalghani, M.R.; Solanki, S.K.; Solanki, J. Optimal coordinated charging and routing scheme of electric vehicles in distribution grids: Real grid cases. *Sustain. Cities Soc.* **2021**, *73*, 103081. [CrossRef]
14. Yao, L.; Lim, W.H.; Tsai, T.S. A real-time charging scheme for demand response in electric vehicle parking station. *IEEE Trans. Smart Grid* **2017**, *8*, 52–62. [CrossRef]
15. Zhao, S.; Lin, X.; Chen, M. Peak-minimizing online EV charging. In Proceedings of the 2013 51st Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 2–4 October 2013; pp. 46–53.
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; Bradford Book: Cambridge, MA, USA, 2018.
17. Wan, Z.; Li, H.; He, H.; Prokhorov, D. Model-free real-time EV charging scheduling based on deep reinforcement learning. *IEEE Trans. Smart Grid* **2019**, *10*, 5246–5257. [CrossRef]

18. Mocanu, E.; Mocanu, D.C.; Nguyen, P.H.; Liotta, A.; Webber, M.E.; Gibescu, M.; Slootweg, J.G. On-line building energy optimization using deep reinforcement learning. *IEEE Trans. Smart Grid* **2019**, *10*, 3698–3708. [CrossRef]
19. Lecun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [PubMed] [CrossRef]
20. Zhou, H.; Jiang, K.; He, S.; Min, G.; Wu, J. Distributed Deep Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet-of-Vehicles. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 9595–9606. [CrossRef]
21. Zhou, H.; Jiang, K.; He, S.; Min, G.; Wu, J. Federated Distributed Deep Reinforcement Learning for Recommendation-enabled Edge Caching. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 9619–9630. [CrossRef]
22. Zhou, H.; Wang, Z.; Zheng, H.; He, S.; Dong, M. Cost Minimization-Oriented Computation Offloading and Service Caching in Mobile Cloud-Edge Computing: An A3C-Based Approach. *IEEE Trans. Netw. Sci. Eng.* **2023**, *10*, 1326–1337. [CrossRef]
23. Liu, K.; Liao, W. Intelligent offloading for multi-access edge computing: A new actor-critic approach. In Proceedings of the 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 7–11 June 2020; pp. 1–6.
24. Song, S.; Fang, Z.; Zhang, Z.; Chen, C.; Sun, H. Semi-online computational offloading by dueling deep-Q network for user behavior prediction. *IEEE Access* **2020**, *8*, 118192–118204. [CrossRef]
25. Zhang, F.; Yang, Q.; An, D. CDDPG: A deep reinforcement learning-based approach for electric vehicle charging control. *IEEE Internet Things J.* **2021**, *8*, 3075–3087. [CrossRef]
26. Da Silva, F.L.; Nishida, C.E.H.; Roijers, D.M.; Costa, A.H.R. Coordination of electric vehicle charging through multiagent reinforcement learning. *IEEE Trans. Smart Grid* **2020**, *11*, 2347–2356. [CrossRef]
27. Neely, M.J. Stochastic network optimization with application to communication and queueing systems. *Synth. Lect. Commun. Netw.* **2010**, *3*, 1–211.
28. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, O.P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Proc. Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390.
29. Diao, X.; Zheng, J.; Cai, Y.; Wu, Y.; Anpalagan, A. Fair data allocation and trajectory optimization for UAV-Assisted mobile edge computing. *IEEE Commun. Lett.* **2019**, *23*, 2357–2361. [CrossRef]
30. Hu, Q.; Cai, Y.; Yu, G.; Qin, Z.; Zhao, M.; Li, G.Y. Joint offloading and trajectory design for UAV-enabled mobile edge computing systems. *IEEE Internet Things J.* **2019**, *6*, 879–1892. [CrossRef]