Digital Communications and Networks(DCN)

HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/dcan

Service chain caching and task offloading in two-tier UAV-assisted vehicular edge computing networks: an Attention-DRL method

Fengyan Wu[a], Chao Yang [*a,d], Yanqun Tang[b], Zhen Li[c] and Shengli Xie[a,d], *Fellow, IEEE*

[a] *School of Automation, Guangdong University of Technology and Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China*
[b] *School of Electronics and Communications, Sun Yat-Sen University, Shenzhen 518107, China*
[c] *Department of Mathematics and Information Technology, The Education University of Hong Kong, HongKong, China*
[d] *The Guangdong-HongKong-Macao Joint Laboratory for Smart Discrete Manufacturing and the Joint International Research Laboratory of Intelligent Information Processing and System Integration of IoT, Guangzhou, China.*

Abstract

Unmanned Aerial Vehicle (UAV)-assisted Vehicular Edge Computing Networks (VECNs) have emerged as a promising solution to enhance service quality for ground vehicle users. However, the growing demands from users and the limited computing and storage resources of UAVs present significant challenges in designing an efficient edge service caching scheme to minimize latency. Moreover, the integration of service caching and task offloading complicates the support of complex tasks by a single UAV. To address these challenges, this paper proposes a novel two-tier UAV-assisted VECNs framework. In this framework, multi-rotor UAVs function as hovering nodes for computational offloading, while a fixed-wing UAV serves as a mobile auxiliary cloud platform, forming a cohesive UAV group. User tasks are structured into a task chain based on the available UAVs. We integrate a joint service chain caching and task offloading scheme that considers UAV computing and storage capacities, duplicate caching, and dynamic transmission latency. To optimize task chain completion latency, we propose an Attention-based Multi-Agent Deep Q-Network (A-MADQN) algorithm. This algorithm incorporates an attention mechanism to narrow the UAV selection space, enabling the selected UAVs to collaboratively make caching and task offloading decisions. Numerical results demonstrate that the proposed algorithm significantly enhances system processing efficiency and reduces task completion latency compared to the benchmark approaches.

KEYWORDS:
Service caching, Task chain, Multi-UAV, Attention mechanism, Multi-agent DQN.

1. Introduction

With the rapid expansion of mobile communication and AI technologies, data generated by connected vehicles has surged significantly. Vehicular Edge Computing Networks (VECNs) offer a promising solution by distributing computation, communication, and caching resources at the network edge node, such as Road Side Unit (RSU), and nearby/parking vehicles. This approach reduces data transmission distances, enabling direct support for computation-intensive and latency-sensitive tasks, thereby enhancing the performance of Intelligent Transportation Systems (ITS) [1, 2]. However, the emergence of various applications has significantly increased the complexity and dynamism of vehicle users' computation tasks. Deploying RSUs extensively along roadways is extremely costly and does not align with dynamic requirements. A cloud-edge collaborative approach [3, 4, 5] that distributes task execution across multiple edge nodes and the core cloud center enhances the overall processing capacity [6, 5]. However, the inherent physical limitations and limited flexibility in this approach can reduce the performance of VECNs.

Unmanned Aerial Vehicle (UAV) has become a crucial component in VECNs, particularly in congested traffic areas, due to their rapid mobility, cost-effectiveness and Line-of-Sight (LoS) communication capabilities. Acting as flying base stations and relay nodes, UAVs support RSUs in meeting the sudden surge in vehicle user demand. Reference [7] explored the use of UAVs as relays to connect uncovered users to nearby base stations after a sudden malfunction. UAVs can also function as mobile nodes to create real-time radio maps for covered ground users [8]. In [9], the authors considered a UAV-enabled wireless-powered mobile-edge computing system, where UAVs assist with computation offloading and energy supplementation for ground users. To maximize the potential and counteract the limitations of individual UAVs, such as limited energy capacity and communication range, transitioning from isolated UAV operations to a collaborative multi-UAV framework is essential [10, 11]. It enables the formation of a UAV cluster to deal with tasks and share resources with RSUs or the ITS cloud center [12, 13, 14, 15, 16]. Additionally, UAVs with varying computing, storage, and communication capabilities can be utilized effectively. For instance, fixed-wing UAVs offer superior computing power and flying range but lack the hovering ability of multi-rotor

*Chao Yang (Email: yangchaoscut@aliyun.com).
[1]Fengyan Wu (Email: 2112204070@mail2.gdut.edu.cn).
[2]Yanqun Tang (Email: tangyq8@mail.sysu.edu.cn).
[3]Zhen Li (Email: lzhen@eduhk.hk).
[4]Shengli Xie (Email: shlxie@gdut.edu.cn).

UAVs. There is growing interest in utilizing diverse UAV types to create UAV-assisted VECNs.

Edge caching is an efficient method to maximize UAVs' limited storage capacity by optimizing and storing contents to meet the user demands in the densely deployed networks [17]. As emerging applications in VECNs increase the complexity of the computation tasks, service caching reduces task completion latency by storing relevant database or program data at edge nodes [18, 19]. Tasks can only be offloaded to edge nodes configured with the appropriate services. Consequently, task offloading and service caching should be jointly optimized, specifically focusing on the strategic decisions regarding task allocation and offloading locations. Existing works have focused on the single task or simple task processing. However, most emerging complex applications involve various dependent execution and storage components. For example, the video surveillance application [20] involves sequential, interdependent tasks such as image capture, object detection, data processing, and decision-making. These tasks cannot be processed by a single edge node or completed within a single time slot.

In this paper, we design an efficient service caching and task offloading scheme in a UAV group-assisted VECNs architecture, that both the fixed-wing and multi-rotor UAVs are considered. There are three main problems should be addressed. 1)*How to design a multiple UAVs-assisted VECNs architecture to maximize the effectiveness of UAVs?* The fixed-wing and multi-rotor UAVs must be strategically deployed to meet vehicle users ' requirements on the road. 2) *How to optimally select UAVs for placing the service chains for the complex task?* For the complex task is divided into a task chain with a set of tasks, the UAVs should be optimally selected to perform the service chain caching. 3) *How to design a joint task offloading and service caching scheme to minimize the task completion latency?* Task offloading and service caching decisions must be optimized, considering the mobility of UAVs and vehicle users, to enhance computation efficiency [21, 22].

Recognizing these problems, we propose a joint service chain caching and task offloading scheme for the complex task in a two-tier UAV-assisted VECNs architecture. Service chain caching and task offloading are closely related, as caching necessary service applications and data on UAVs significantly reduces the latency and bandwidth costs associated with offloading computation tasks. This ensures faster and more efficient task execution by providing immediate access to required services on the UAVs.

The multi-rotor UAVs hover to provide computation and communication services for the covered vehicle users on the road, and a fixed-wing UAV supports additional requests by flying among the hovering UAVs and vehicles. For the vehicles, the complex task is divided into a task chain with a set of tasks. Combined with duplicate and centralized caching, service chain caching is proposed to balance the demands of computational efficiency and resource utilization. Correspondingly, serial and parallel task offloading schemes are developed to maximize the utility of UAVs. In order to reduce the searching space, an Attention-based Multi-Agent Deep Q Network (A-MADQN) algorithm is proposed to efficiently determine the optimal task offloading and service caching decisions. The main contributions are summarized as follows:

- We develop a two-tier UAV-assisted VECNs architecture for the complex tasks of vehicle users, where multi-rotor UAVs primarily handle computation offloading and a fixed-wing UAV serves a supplementary role in the network, functioning as a mobile cloud platform.
- We propose an efficient joint service chain caching and task offloading scheme to minimize the task completion latency. The complex task is divided into a task chain, with serial and parallel task offloading patterns considered. Multiple UAVs are selected for service chain caching to support the task chain's execution.
- We design an attention-based MADQN algorithm to utilize the limited resources of UAVs. The attention mechanism determines the offloading weight for each edge node, feeding into the MADQN method to reduce the optimal search space and enhance the collective decision-making process for efficient computation task distribution.

In addition, we also evaluate the impacts of key parameters through extensive simulations, demonstrating the superiority of our proposed algorithm compared to benchmark solutions. The remainder of this article is organized as follows. In Section II, we review the previous related works. Section III introduces the system model and computation offloading, and Section IV describes the service caching and problem formulation. The attention-based MADQN algorithm is provided in Section V, and numerical results are discussed in Section VI. Finally, Section VII concludes the work.

2. Related work

In this section, we describe the multi-UAV networks, the existing joint service caching and task offloading algorithms, and then present a brief review of these works.

2.1. Multi-UAV networks

UAVs, with their mobility, LoS communications, and flexible deployment, can function as temporary aerial base stations or edge computing platforms to serve ground vehicles, particularly in the congested roads [23]. To enhance the UAV efficiency, the task offloading division ratio, UAV deployment, communication, and computation resource allocation [9, 24, 25] were analyzed to reduce latency and energy consumption. The resource management in UAV-assisted VECNs was further optimized via multi-agent reinforcement learning algorithm in [12]. In a UAV-aided IoT network, the ultra-reliable low-latency computation offloading problem was studied to maximize the transmission rates [26]. However, the computation and storage capacity of a single UAV is limited, making it difficult to handle complex tasks and larger-scale operations. A viable and increasingly adopted solution is the integration of cloud computing platforms, which serve as a supplemental resource to enhance the capabilities of UAVs. In [27], the authors proposed an online optimal computation offloading and multi-hop routing scheme in an edge-cloud environment. While cloud centers provide additional computing resources, the distance from end-users results in prolonged transmission and communication delays, adversely impacting system performance. In this paper, we propose a two-tier UAV-assisted network architecture. It integrates a lower layer of multi-rotor UAVs as primary edge nodes, and an upper layer of a fixed-wing UAV as a supplementing node. It maintains closer proximity to end-users than traditional cloud platforms inherently.

2.2. Joint service caching and computation offloading

An efficient and suitable task offloading strategy can accelerate computation and extend the battery life of terminal devices. Subsequent studies have explored various aspects of the task offloading problem, such as resource utilization [28], delay minimization [29], and overall system consumption optimization [30, 31]. For complex tasks, the task processing needs the necessary program data, which affects the specific edge node selection. Service caching can address the issue of limited storage resources in edge nodes by pre-storing applications and data [32, 33]. Consequently, there is a growing focus on the joint optimization of the service caching and task offloading. In [34], a pricing strategy was introduced for edge nodes that integrates with service caching decisions to guide the offloading actions of wireless devices in cellular networks. In [35], a Mixed Integer NonLinear Programming (MINLP) model was presented, and the service caching placement, computation offloading, and resource allocation were optimized. In [36], the offloading problem was addressed within the context of fixed service caching, neglecting the dynamic aspect of cache updating on the server. In fact, service caching data should be timely updated to align with the ongoing task processing, especially for the dependent

tasks. In [37], a dependency-aware task offloading algorithm was studied. Furthermore, an efficient algorithm based on convex programming was designed to minimize the completion time of the application in [38].

Unlike the existing works that predefined the dependency relationship of tasks and cannot handle dynamic resource scenarios, we consider a service chain caching and task offloading scheme for the complex task in a two-tier UAV-assisted VECNs. The complex task can be divided into a task chain based on the varying edge computing environments. Additionally, we emphasize the importance of service chain caching, focusing on its constraints and the necessity of timely updates, to ensure that all tasks of task chain can obtain the necessary support and resources for effective execution, enhancing the overall system performance and efficiency.

3. System model

In this section, we introduce the system model of a two-tier UAV-assisted VECNs. Particularly, we present the corresponding network architecture, including the attention-based task chain model, computation and communication models. The key parameters and notations are summarized in Table 1.

Table 1
NOTATION

Notation	Definition
N	Total number of multi-rotor UAVs
M	Types of services
S	Task chain
K	Length of the task chain
$E_i^{'}$	The available energy of the UAV i
$Z_i^{'}$	The storage space of the UAV i
s_k	Tasks in the task chain
d_k	The computation input data size of task s_k
c_k	The requested CPU cycles of task s_k
τ_k	The tolerance latency maximum of task s_k
f_k	The required service of task s_k
e_{f_k}	The energy required for the cache service f_k
z_{f_k}	The size of service f_k
ψ_{f_k}	The caching probability of service f_k
μ_k	The task division decision of task s_k
w_k	The task division weight of task s_k
α_k	The computation offloading decision of task s_k
$\beta_{i,k}$	The service caching decision of service f_k in UAV i

3.1. Two-tier UAV-assisted VECNs

As shown in Figure 1, the two-tier UAV-assisted VECNs consists of a ground user layer and a two-tier UAV edge layer, which can communicate with each other through wireless communication links. On congested roads, the ground users simultaneously generate computation requests with a high computing burden. Additionally, there are two varieties of UAVs in the two-tier UAV edge layer. Multi-rotor UAVs, deployed near ground users, serve as primary edge execution nodes due to their flexibility and maneuverability. With significantly greater computation capacity than ground users, these UAVs efficiently handle computationally intensive tasks. Despite their limited cache and computation capacities, they reduce the processing burden on ground users, enhancing overall system performance. Meanwhile, the fixed-wing UAV, with extensive coverage and abundant resources, operates in the upper layer, flying among hovering UAVs to facilitate task offloading for the covered vehicle users.

The two-tier UAV system is motivated by the differing coverage ranges of these UAVs. The fixed-wing UAV can cover extensive areas, providing overarching support and efficient resource management across wide regions. Conversely, multi-rotor UAVs, with their localized operation capabilities, can respond swiftly to immediate, nearby user requests. This distinction enables the system to more effectively manage time-varying task demands, ensuring that resources are dynamically allocated according to real-time needs.

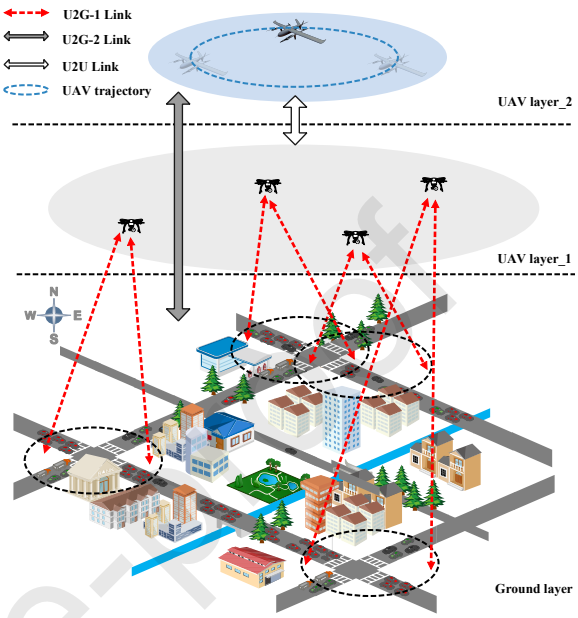


Fig. 1. Two-tier UAV-assisted VECNs system model.

Ground users generate complex tasks that are divided into dependent service modules, forming a task chain that requires sequential execution. It requires a specific order for execution, the tasks in the chain depends on the output of the previous tasks. The whole system operates in a slot-by-slot fashion. At the beginning of each slot, the vehicle users generate tasks and form a completed task. The multi-rotor UAVs are selected to form a UAV group, perform the service caching and task offloading for the users. Based on the UAV group, the original sequential task chain becomes a serial-parallel heterogeneous chain, and be executed with a collaboration calculation model.

3.2. Attention-based task chain model

The task chain formulation is linked to the UAV group, making attention-based UAV selection an effective method for identifying suitable UAVs. Consider a task chain consisting of multiple dependent serial tasks, denoted as $S = \{s_1, \ldots, s_k, \ldots, s_K\}$, where s_k is a task k of the task chain, and K is the number of tasks. The task can be described as $s_k = (d_k, c_k, \tau_k, f_k)$, where d_k represents the computation input data size (in bit), c_k indicates the requested CPU cycles, τ_k implies the tolerance latency, and f_k signifies the required service data. Each task requires specific program data and computation resources. The UAVs serve as service caching nodes to process these tasks.

An attention mechanism is introduced to align the tasks with the optimal computation and storage resources of UAVs. It evaluates each task's requirements and calculates an attention score for each multi-rotor UAV based on three key factors: the alignment of the UAV's cached service data with the task requirements, the communication quality between the multi-rotor UAV and the vehicle user, and the UAV's capabilities, including computation resources, energy consumption, and battery life. Then, a subset of UAVs is chosen to form the UAV group.

Actually, when the number of UAVs is larger, part of UAVs can select duplicate caching to reduce the computation time. The special

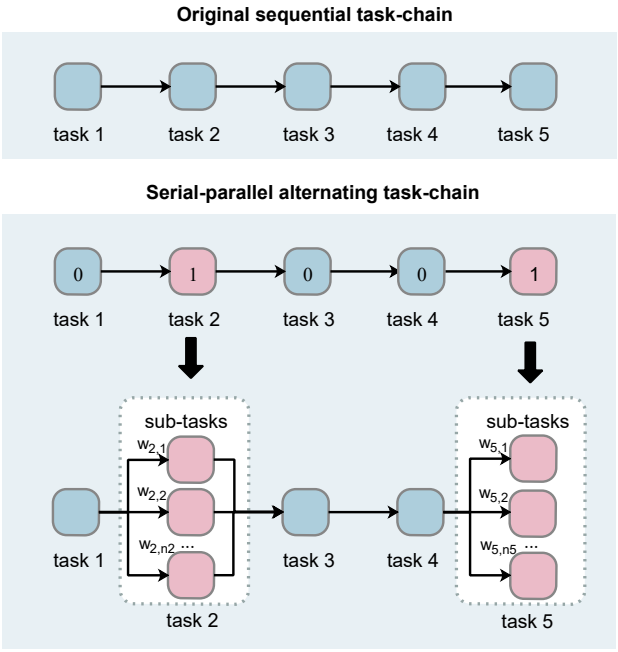


Fig. 2. Depiction of the task division process.

task can be divided into sub-tasks that can be executed simultaneously, thereby utilizing the computational resources of the multi-rotor UAVs more effectively. Consequently, the task chain evolves into a heterogeneous combination of serial and parallel processes, enhancing network resource utilization and efficiency. The task division process is shown in Fig. 2. Specially, the attention scores can be normalized to the corresponding task division weights w_k, which reflects the relative importance of the overall tasks.

3.3. Computation and communication model

The model assumes a reliable communication environment with minimal interference and signal loss. This assumption allows us to concentrate on the computational and caching aspects of the system, avoiding the complexities introduced by fluctuating communication conditions.

As shown in Figure 1, each ground vehicle user can execute its computation task locally or offload part of subtasks in the task chain to a multi-rotor UAV or the fixed-wing UAV. Denote $\alpha_k = \{0, 1, 2\}$ as the offloading decision of task s_k in vehicle user k, as:

$$\alpha_k = \begin{cases} 0, & \text{local computing} \\ 1, & \text{the fixed-wing UAV computing} \\ 2, & \text{attention group computing} \end{cases} \tag{1}$$

Specifically, $\alpha_k = 0$ if ground vehicle user k can complete its task locally, $\alpha_k = 1$ if user k chooses the fixed-wing UAV for remote execution, and $\alpha_k = 2$ when user k offloads its task to the attention UAV group. Then we define the offloading correlation factor as $\alpha_{k,l}$, $\alpha_{k,f}$ and $\alpha_{k,u}$. Specifically, we have

$$\begin{cases} \alpha_{k,l} = 1, \alpha_{k,f} = 0, \alpha_{k,u} = 0 & \text{if } \alpha_k = 0 \\ \alpha_{k,l} = 0, \alpha_{k,f} = 1, \alpha_{k,u} = 0 & \text{if } \alpha_k = 1 \\ \alpha_{k,l} = 0, \alpha_{k,f} = 0, \alpha_{k,u} = 1 & \text{if } \alpha_k = 2 \end{cases} \tag{2}$$

where $\alpha_{k,l} = 1$ indicates that the task will be executed locally, otherwise $\alpha_{k,l} = 0$. Likewise, $\alpha_{k,f} = 1$ indicates that the task will be executed at the fixed-wing UAV, otherwise $\alpha_{k,f} = 0$. And $\alpha_{k,u} = 1$ indicates that the task will be executed at the multi-rotor UAV attention group, otherwise $\alpha_{k,u} = 0$.

1) *Local computing*

Assuming each ground user possesses all necessary program data for task processing, the local processing time becomes solely the computation time, as follows:

$$t_k^l = \frac{c_k}{c_k^l} \tag{3}$$

where c_k^l denotes the local CPU frequency and is constrained by a maximum frequency $c_k^l \le c_{max}$, c_k is the total number of CPU cycles required for computing task s_k.

2) *Fixed-wing UAV computing*

Accordingly, the uplink transmission delay from ground user k to the fixed-wing UAV is given by:

$$t_k^{f,tran} = \frac{d_k}{r_{k,f}} \tag{4}$$

where $t_k^{f,tran}$ is the uplink transmission, d_k is the input data size, and $r_{k,f}$ is the achieved uplink transmission rate from user k to the fixed-wing UAV, expressed as follows:

$$r_{k,f} = B_{k,f} \log_2 \left(1 + \frac{p h_{k,f}}{\sigma^2}\right) \tag{5}$$

where $B_{k,f}$ is the communication bandwidth, p is the transmission power of ground user in the upload link, and $h_{k,f}$ is the LoS channel gain, which is assumed constant but changes from the boundary of each offloading period. σ^2 is the power spectral density of Gaussian noise. For LoS communications, the OFDMA is adopted to avoid the channel interference [39].

The computation delay for task s_k on the fixed-wing UAV can be derived by:

$$t_k^{f,com} = \frac{c_k}{c_k^f} \tag{6}$$

where c_k^f denotes the CPU frequency of the fixed-wing UAV, c_k is the total number of CPU cycles required for computing task s_k.

For the task execution results are generally much smaller than the input data. Set that the transmission time of the execution results are ignored, the total delay for a user offloading task s_k to the fixed wing UAV is as follows:

$$t_k^f = t_k^{f,tran} + t_k^{f,com} = \frac{d_k}{r_{k,f}} + \frac{c_k}{c_k^f} \tag{7}$$

3) *Attention group computing*

The UAV group computing delay is associated with the number of UAVs. With more UAVs available, the tasks can be processed in parallel, reducing the overall processing time. The time delay t_k^u for task s_k in multi-rotor UAV group is:

$$t_k^u = \begin{cases} t_k^{u,com} + t_k^{u,tran}, & \mu = 0 \\ \max\left(w_{k,1} t_{k,1}^u, \ldots, w_{k,u_{i,k}} t_{k,u_{i,k}}^u\right), & \mu = 1 \end{cases} \tag{8}$$

When $\mu = 0$, the task s_k is processed on a single multi-rotor UAV. The total delay cost to execute the computation task s_k is described as the sum of computation delay and transmission delay, as given by:

$$\begin{aligned} t_k^u &= t_k^{u,com} + t_k^{u,tran} \\ &= \frac{c_k}{c_k^u} + \frac{d_k}{r_{k,u}} \end{aligned} \tag{9}$$

where c_k^u denotes the CPU frequency of the multi-rotor UAV in the attention group, and c_k is the total number of CPU cycles required for s_k, $r_{k,u}$ is the uplink transmission rate, similar as Eq. (5).

When $\mu = 1$, the task s_k is divided and processed in parallel across multiple multi-rotor UAVs. The total delay for executing the task s_k is the maximum delay of these parallel sub-tasks, as:

$$t_k^u = \max\left(w_{k,1} t_{k,1}^u, w_{k,2} t_{k,2}^u, \ldots, w_{k,u_{i,k}} t_{k,u_{i,k}}^u\right) \tag{10}$$

where $w_{k,u_{i,k}}$ is the task division weight of the multi-rotor UAV i, for the task division is based on the normalized attention scores w_k, $t_{k,u_{i,k}}^u$ is the delay of subtask i executed on the multi-rotor UAV i.

4. Service chain caching strategy

In the two-tier UAV-assisted VECNs, UAVs can perform task processing only if the relevant service data has been deployed beforehand. The service caching strategy directly impacts system performance, where the binary variable $\beta_{i,k}$ is the service caching decision for service f_k on multi-rotor UAV i, as given by:

$$\beta_{i,k} = \{0, 1\} \tag{11}$$

where $\beta_{i,k} = 1$ if service f_k is cached, otherwise $\beta_{i,k} = 0$.

Two key factors are crucial for optimizing caching decisions for multi-rotor UAVs: node effectiveness and service caching probability. In the former, it refers to the UAV's ability to store services, considering access speed and storage capacity. In the latter, it directs the UAVs to prioritize storing services with higher demand probabilities. The calculation of these caching parameters is described as follows.

Node effectiveness. This parameter assesses a node's capacity for caching operations, factoring in available caching space and energy. In particular, a node must ensure it has adequate storage and energy before accepting a service cache request.

Let $\rho_{i,k}$ be the effectiveness of the multi-rotor UAV i receiving service f_k, as:

$$\rho_{i,k} = \begin{cases} 0, & E_i^{'} < e_{f_k} \\ 1, & E_i^{'} \geq e_{f_k} \text{ and } Z_i^{'} \geq z_{f_k} \\ (0,1), & E_i^{'} \geq e_{f_k} \text{ and } Z_i^{'} < z_{f_k} \end{cases} \tag{12}$$

where z_{f_k} is the size of service f_k, e_{f_k} is the energy required for the cache service f_k, $Z_i^{'}$ and $E_i^{'}$ denote the storage space and the available energy of the UAV i.

In details, when a UAV lacks sufficient energy to cache service f_k, $\rho_{i,k} = 0$. Conversely, if UAV i possesses both adequate storage space and energy for service f_k, $\rho_{i,k} = 1$. However, if the UAV i has sufficient energy but lacks adequate storage space for service f_k, $\rho_{i,k}$ is assigned a probabilistic value between 0 and 1, reflecting the probabilistic nature of the caching capability.

Service caching probability. It can be calculated at each node by determining the number of requests received for service f_k, req_{f_k}, over the maximum number of requests req_{f_k}, as:

$$\psi_{f_k} = \frac{req_{f_k}}{\max(req_{f_k})} \tag{13}$$

The storage system uses a Least Frequently Used (LFU) policy, indicating that when the service needs to be updated and there is insufficient capacity, the least accessed service will be purged. We denote the probability of cache update $\xi_{i,k}$ as:

$$\xi_{i,k} = \frac{\min(fre_{f_{i,k}})}{\sum(fre_{f_i})} \tag{14}$$

where $\min(fre_{f_{i,k}})$ is the frequency of the least accessed service, $sum(fre_{f_i})$ is the total number of accessed services. To cache a new service, the LFU policy is combined with the Cache Hit Ratio (CHR) for eviction decisions.

Service caching decision. Upon receiving a data packet for service f_k, UAVs evaluate the node's effectiveness and service caching probability to determine whether to cache the data. The node must first be deemed effective before caching is performed based on the service cache probability.

Service caching decisions incorporate dynamic factors related to both demands (i.e., service caching probability ψ_{f_k}) and supply (i.e., effectiveness of the node $\rho_{i,k}$) for a more optimized caching results. The service caching decision β_k is expressed as follows:

$$\beta_k = \begin{cases} \psi_{f_k}, & \rho_{i,k} = 1 \\ \xi_{i,k} \cdot \psi_{f_k}, & \rho_{i,k} = (0,1) \\ 0, & \rho_{i,k} = 0 \end{cases} \tag{15}$$

Specifically, the UAV performs service caching based on the service cache probability when $\rho_{i,k} = 1$. On the other hand, the UAV does not perform caching operation if $\rho_{i,k} = 0$. However, in scenarios where $\rho_{i,k}$ is set to a probabilistic value between 0 and 1, the final caching decision is based on the product of the service cache probability and the cache update probability.

In this paper, we propose an integrated framework for service chain caching and task offloading in a two-tier UAV-assisted VECNs. The objective is to optimize cooperative service chain caching and task offloading decisions to minimize chain completion delay. The optimization problem is formulated as follows:

$$\begin{aligned} \boldsymbol{P}: \quad & \min \sum_{k=1}^{K} a_{k,l} t_k^l + a_{k,f} t_k^f + a_{k,u} t_k^u \\ s.t. \quad & C1: \ a_{k,l}, a_{k,f}, a_{k,u} \in \{0,1\} \\ & C2: \ a_{k,l} + a_{k,f} + a_{k,u} = 1 \\ & C3: \ t_k \leq \tau_k \\ & C4: \ T_k + t_k \leq T_{k+1} \\ & C5: \ E_i^{'} < E_{iM}^{'} \\ & C6: \ E_i^{'} < E_{iB}^{'} \\ & C7: \ Z_i^{'} < Z_{iM}^{'} \\ & C8: \ \beta_{i,k} \in \{0,1\} \\ & C9: \ \mu_k \in \{0,1\}, w_k = [0,1] \end{aligned} \tag{16}$$

where $C1$ and $C2$ indicate that the task must be selected for only one type of execution destination. $C3$ represents that the task must be completed within the completion delay. $C4$ is the task dependency constraint, which states that the start time T_{k+1} of each subsequent task s_{k+1} must occur only after the completion of the preceding task s_k. $C5$ and $C6$ constraints ensure that the energy consumption of each UAV does not exceed the available energy and the UAV's maximum battery capacity, respectively. $C7$ is cache storage space constraints. $C8$ represents the service caching decision of service f_k in multi-rotor UAV i. $C9$ is the task division decision and task division weight. Actually, we do not consider the task processing waiting time of the task chain, for the UAVs have enough communication and computation resources to address the received tasks.

5. Attention-based multi-agent DQN algorithm

The primary challenge in solving problem $\boldsymbol{P}$ is that the service caching and task offloading decisions are coupled with each other. Based on the system model, we propose an attention-based MADQN (A-MADQN) algorithm to enhance decision-making effectiveness in the complex and dynamic environments. To be more precise, we first detail MADQN and the attention mechanism separately, and then outline the framework of the proposed A-MADQN algorithm.

5.1. Multi-agent deep Q-network

For problem **P**, the service chain caching and task offloading are coupled with each other. As the number of vehicle users increases, the computational complexity grows, making traditional optimization techniques insufficient for rapid problem-solving. DRL combines neural networks with Q-learning principles to provide a robust decision-making framework. Two pivotal technologies, such as experience replay and target networks, enhance the stability and learning efficiency of the DQN model. Set that there are N multi-rotor UAVs, and each multi-rotor UAV is an agent. In the multi-agent DQN, agents adopt the way of Independent Q-Learning (IQL), where each agent ignores other agents of the environment and learns its own strategy, thus minimizing system latency and resource consumption.

State space. Each agent observes the state information, including the current task state, the multi-rotor UAV state, and distances. In the paper, we define the system states as $S = \{\boldsymbol{x}, \boldsymbol{y}, d\}$. $\boldsymbol{x} = \{x_d, x_c, x_\tau, x_f, x_z, x_s, x_a\}$ denotes the current task state. This encompasses the computation input data size x_d, the requested CPU cycles x_c, the maximum tolerance

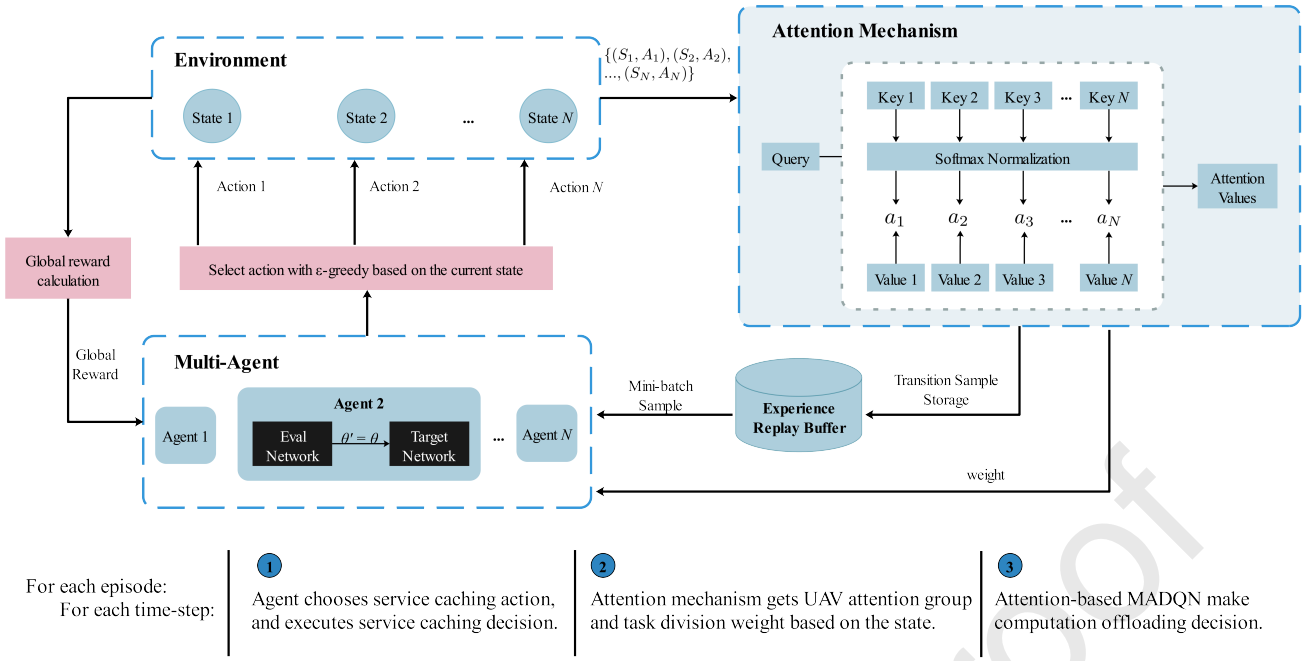


Fig. 3. The framework of the attention-based MADQN algorithm

latency of the task x_τ, and the required service. The required service is further detailed by its service ID x_f, service size x_z, cache status x_s, and access frequency x_a. $\boldsymbol{y} = \{y_c, y_f, y_z, y_b\}$ denotes the multi-rotor UAV state, i.e., the CPU frequency y_c, the cached service y_f, the cache space y_z, and battery y_b. d denotes the distance between the multi-rotor UAV and ground user generating the current task.

Action space. Upon receiving the state information, each agent decides which services to cache. In this paper, we consider that there are N multi-rotor UAVs and M service types. The service caching decision variable is $\beta_{i,k} = \{0, 1\}$, and then the action space of service caching is 2^{MN}. In this case, with the increasing number of multi-rotor UAVs and service types, the action space expands exponentially, consequently complicating the model training process. To address this issue, we have redefined the action space.

Set $A = \{a_1, \ldots, a_i, \ldots, a_N\}$ as the action space, where a_i ranging from 0 to $2^M - 1$. In the defined action space, actions are designated as integer values, each uniquely corresponding to a subset of service caching. These integer actions are then decoded into binary form to specify the caching decisions for each service.

Reward. The reward quantifies the outcome of an action, calculated based on the changes in the external environment. According to objective function in problem **P**, the system latency in computation offloading process is defined as the system reward.

The shared global reward R is calculated, and be linked to the actions of all agents. It is uniformly applied in the learning process of every agent within the MADQN framework. This approach fosters cooperative behaviors among agents, improving both individual performance and overall task support. Additionally, we include a penalty δ in the reward calculation when the multi-rotor UAV lacks sufficient power or computing resources to perform the received task.

Then, the reward of agent i processing task s_k, $r_{i,k}$ is:

$$t_{i,k} = a_{k,l} t^l_{i,k} + a_{k,f} t^f_{i,k} + a_{k,u} t^u_{i,k} \tag{17}$$

$$r_{i,k} = \frac{1}{t_{i,k}} - \delta_{i,k} \tag{18}$$

The training process of the multi-agent DQN model is described in two distinct phases. In the experience collection phase, the agent gathers experience by interacting with the environment. It observes the current state, inputs this into the evaluation network and obtains Q-values for each possible action. The agent selects actions using an epsilon-greedy strategy, starting with random choices and increasingly opting for actions with the highest Q-values. Following each action, the environment responds with a reward and transition to a new state. The resulting tuple (s, a, r, s') from each interaction is stored in the experience replay pool for future learning. Once a predetermined number of interactions are reached, the agent enters the training phase. The agent randomly samples a batch of experiences from the pool, inputting the state s into the evaluation network and the new state s' into the target network to obtain the corresponding q_{eval} and q_{target}. The calculated loss for this batch is utilized for gradient descent optimization to train the network. After training, the cycle reverts to the first phase.

In this paper, the training process adopts a combination of centralized education and decentralized implementation. Each agent independently selects actions based on its policy and, after all agents have acted, they receive a shared global reward. Agents interact with the environment to gather experiences, which are stored in individual buffer pools. Each agent randomly selects experiences from its buffer pool to train its network.

5.2. Attention mechanism

To reduce the optimal search space, the attention mechanism assigns offloading weights to each edge node, and integrating this information feeds into the MADQN method. This approach enables the model to dynamically prioritize relevant input data, enhancing task execution accuracy and efficiency, particularly in complex environments with numerous influencing variables. The attention mechanism enhances task execution by prioritizing key features, improving both accuracy and efficiency.

In the multi-agent setting, where each agent's actions impact the environmental changes and rewards, the attention mechanism improves coordination by enabling agents to focus on the most critical information. This integration enhances collaboration and informs the offloading decision process. This mechanism assigns weights to each agent based on their match with the requirements, reducing computational complexity and enhancing task offloading efficacy.

For multi-rotor UAVs handling task s_k, the attention mechanism evaluates and prioritizes UAVs based on task-specific requirements and UAV capabilities, optimizing task assignment and execution within the multi-rotor UAV swarm.

Algorithm 1 Attention mechanism process

Require: Attributes of multi-rotor UAVs and task requirements.
Ensure: Attention group and attention weights.
1: Randomly initialize feature sets and the query vector.
2: **for** Each multi-rotor UAV in the fleet. **do**
3: Convert attributes into feature representations.
4: Calculate preliminary weights for each feature based on its importance.
5: Apply the softmax function to the preliminary weights to get normalized attention weights
6: Construct context vector for multi-rotor UAV by combining features with their attention weights
7: **end for**

Feature representation: The mechanism begins by converting essential multi-rotor UAV attributes, such as communication quality and computational resources, into a set of features using specific functions that map the raw data into a more processable form.

Attention weight computation: A query vector highlights the importance of each multi-rotor UAV attribute, assigning weights to balance their significance. These weights are adjusted through a learning process to reflect each attribute's relevance to the task.

Softmax normalization: To ensure proportionality and comparability of the calculated weights across all features, maintaining their validity and rationality.

Context vector construction: Finally, the attention mechanism consolidates normalized weights with the corresponding features to construct a context vector for each multi-rotor UAV. This vector encapsulates the UAV's overall capabilities and suitability for the given task, guiding the task assignment process.

The Attention Mechanism is shown in Algorithm 1.

5.3. The framework of the A-MADQN algorithm

The A-MADQN algorithm facilitates efficient collaboration among multiple UAVs within a shared environment. The attention mechanism sharpens focus on critical information, improving decision-making and overall performance. Agents operate independently, processing environmental data to reduce resource consumption and latency, enabling autonomous operations. A shared global reward drives cooperation and adaptability to varying network conditions. This reward system, integrated with the attention mechanism, optimizes both individual and collective agent performance, balancing autonomy and teamwork in complex scenarios. The framework of the proposed A-MADQN algorithm is shown in Fig. 3.

The detailed A-MADQN is demonstrated in Algorithm 2. We implement our proposed A-MADQN algorithm in three major steps.

Step 1: Multi-rotor UAVs select the service caching action set.

Step 2: Task division decisions are made, and task division weights are calculated using the attention mechanism.

Step 3: Tasks are offloaded based on the computation offloading decision and weight values, with the experience gained integrated back into the model for continued learning and training.

6. Simulation results

In this section, we evaluate the performance of the proposed algorithm through numerical simulations. We investigate the effect of different parameter settings on the performance of the proposed scheme and compare it with other baseline schemes.

Algorithm 2 A-MADQN based Service Caching and Computation Offloading Algorithm

Require: Episode length, discount factor, replay buffer size, mini-batch size, number of steps to update the target network.
Ensure: Service caching and computation offloading decision.
1: Randomly generate N multi-rotor UAVs uav, K vehicle users and 1 fixed-wing UAV.
2: Initialize the Q network and target Q network with weights θ and θ' for all agents.
3: **for** episode $l = 0 : episode_{max}$ **do**
4: Initialize system environment.
5: **for** time slot $t = 0 : T$ **do**
6: **for** each agent $i \in N$ **do**
7: Observe state s_i, and choose action a_i:

$$a_i = \begin{cases} \arg\max Q(s_i, a_i; \theta) & \text{with probability 1-}\epsilon \\ \text{random action with probability } \epsilon \end{cases}$$

8: uav_i execute the service caching action a_i.
9: **end for**
10: All uav have taken the service caching actions;
11: Submit all states and actions to Attention Mechanism, and obtain attention group and scores;
12: Determine computation offloading decision;
13: All agents receive global reward r.
14: **for** each agent $i \in N$ **do**
15: Obtain global reward r and the next state s_i';
16: Store transition (s_i, a_i, r, s_i') in experience pool.
17: **if** experience pool is full **then**
18: Randomly sample mini-batch (s_i, a_i, r, s_i') from experience pool;
19: Calculate Q value and target Q value;
20: Optimize error between Q network and target Q network using adaptive moment estimation.
21: **end if**
22: **end for**
23: **if** the current uav run out of energy **then**
24: Introduce a penalty δ, and redo the current task.
25: **end if**
26: **end for**
27: **end for**

6.1. Experimental settings

All the computations are performed in Python with an Intel Core i5-12490F 3.00-GHz CPU and 16 GB of memory. The proposed A-MADQN algorithm is implemented on the PyTorch platform with the Adam optimizer.

Specifically, 40 vehicle users are uniformly and randomly distributed within a $1km \times 1km$ square area, with the deployment of 5 multi-rotor UAVs and 1 fixed-wing UAV. The multi-rotor UAVs operate at 100 m altitude, while the fixed-wing UAV operates at 500 m, respectively. Each vehicle user generates one task per time step, with tasks having specific service requirements. Storage resources for these services range from 2 to 5 GB. Additional simulation parameters are detailed in Table 2. All parameters default to the values specified in Table 2, with settings in each figure taking precedence wherever applicable.

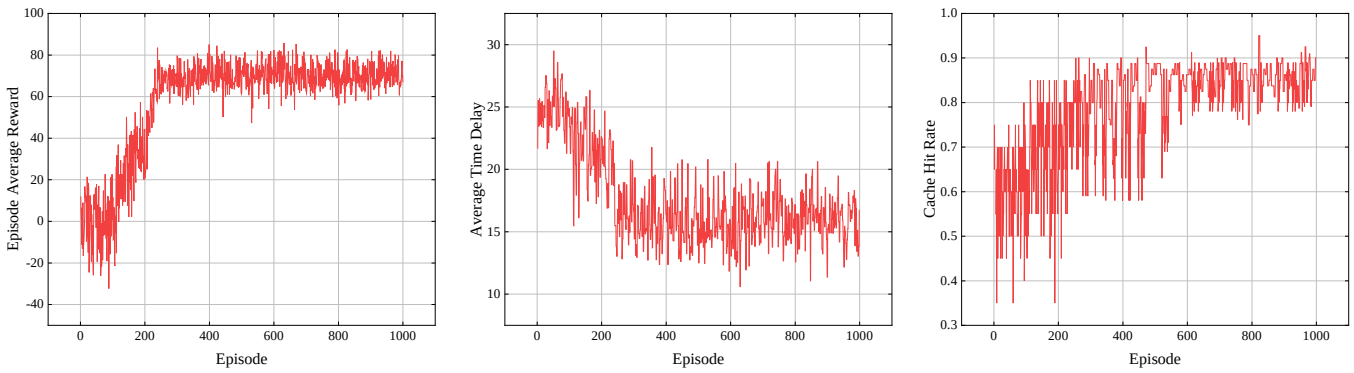
6.2. Convergence analysis

To ascertain the convergence reliability of the proposed A-MADQN method, we investigate a comprehensive research analysis focusing on three key performance indicators: episode average reward, average time delay, and Cache Hit Rate (CHR). Especially, CHR is the primary metric for evaluating the effectiveness of a caching strategy, as follows:

$$CHR = \frac{cache_{hits}}{cache_{req}} \tag{19}$$

where $cache_{hits}$ denotes the number of hits to the service cache, and $cache_{req}$ denotes the number of service requests.

The simulation results are presented in Fig. 4. Fig. 4a illustrates the convergence curves of the episode average reward over 1000 episodes.



(a) Episode average reward. **(b)** Average time delay. **(c)** Cache hit rate.

Fig. 4. Convergence analysis.

Table 2
SIMULATION PARAMETERS

Parameter	Value
Episode	2000
Discount factor	0.9
Learning rate	1e-4
Replay buffer size	2000
Mini-batch size	64
Service number	40
Task number	20
UAVs number	5
User compute CPU	3e8Hz
User transmit power	[0.5, 1.2]W
Multi-rotor uav cache space	[50,100]Mb
Multi-rotor uav battery	[200,400] Kj
Multi-rotor uav compute CPU	[1e9,2e9]Hz
Multi-rotor uav B	[1e5,1.2e5]Hz
Fixed-wing uav B	2e5Hz
Fixed-wing uav compute CPU	9e9Hz
Task data	[5e6,1e7]Bit
Task compute CPU	[5e8,1e9]Hz

As shown in Fig. 4a, the proposed A-MADQN curve initially rises rapidly within the first 200 training episodes and then stabilizes. Fig. 4b shows the average task chain completion time delay for the two-tier UAV-assisted VECNs in delivering network services to vehicle users. The proposed algorithm reduces total delay after 200 episodes and converges to an optimal time delay range, meeting our anticipated standards for service caching and computation offloading in VECN scenarios. Fig. 4c illustrates the convergence of the cache hit rate during the multi-rotor UAV service caching process. The CHR, representing the percentage of requests served by multi-rotor UAVs, increases with episodes, indicating effective service caching by the network.

6.3. Method comparison

The performance of our proposed **Attention-based Multi-Agent Deep Q Network (A-MADQN)** method is evaluated against three benchmark schemes as follows:

1) **Multi-Agent Deep Q Network (MADQN):** The attention mechanism is omitted and the system parameter configurations are kept identical to those in A-MADQN method to ensure a fair comparison.

2) **Random Cache (RC):** Each multi-rotor UAV caches necessary services randomly until its caching capacity is reached.

3) **Greedy Algorithm (Greedy):** This scheme consistently selects the action with the highest Q-value in each time slot, focusing solely on immediate exploitation without balancing exploration.

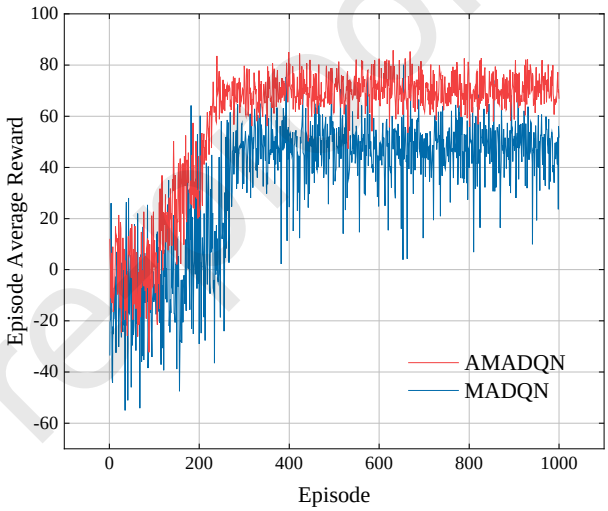


Fig. 5. Convergence of the proposed A-MADQN and MADQN algorithms.

Fig. 5 shows the convergence curves of the average reward for A-MADQN and MADQN as the episodes progress. For A-MADQN, the curve raises quickly and then be stabilized at the 200-th episode. For MADQN, while its performance trend mirrors that of A-MADQN, it exhibits a slower convergence rate, reaching convergence around 300 training episodes, and a smaller reward is obtained. The main reason is that A-MADQN enables network to focus more on valuable information, facilitating the learning of a more effective and attentive cooperation policy. Therefore, A-MADQN demonstrates a significantly faster training speed compared to MADQN, converging and yielding comparatively favorable results.

Fig. 6 illustrates the performance comparison of A-MADQN with other benchmark methods under different numbers of multi-rotor UAVs. As the number of multi-rotor UAVs increases, the time delay decreases, indicating improved performance of the A-MADQN algorithm. A similar trend is observed in the other three methods, primarily due to the increased availability of UAVs for assisting vehicle users with computation offloading. A-MADQN consistently shows optimal performance across different numbers of multi-rotor UAVs, whereas RC performs the worst. This scenario suggests that relying solely on a fixed-wing UAV for computation offloading significantly increases the overall task chain completion. In the proposed A-MADQN algorithm, an interesting performance trend emerges when compared to the Greedy algorithm. Initially, with fewer multi-rotor UAVs, the Greedy algorithm performs worse. However, as the number of multi-rotor UAVs increases, its performance improves. Despite this progress, it still does not achieve the efficiency levels of A-MADQN, which maintains consistent superior-

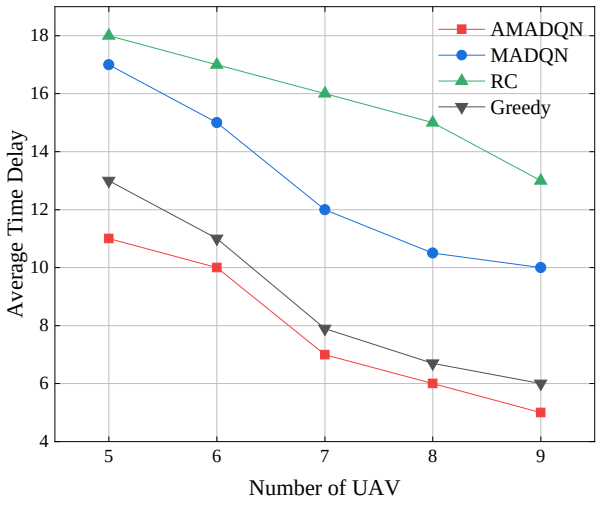


Fig. 6. Performance comparison of A-MADQN and other benchmark methods with varying numbers of multi-rotor UAVs.

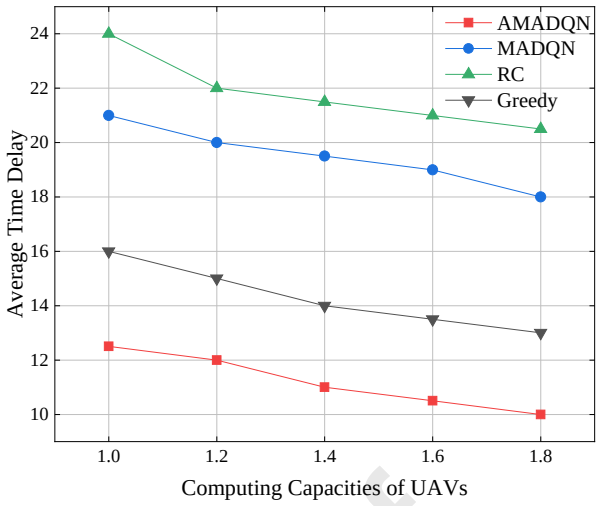


Fig. 8. Performance comparison of A-MADQN and other benchmark methods across varying computing capacities of UAVs.

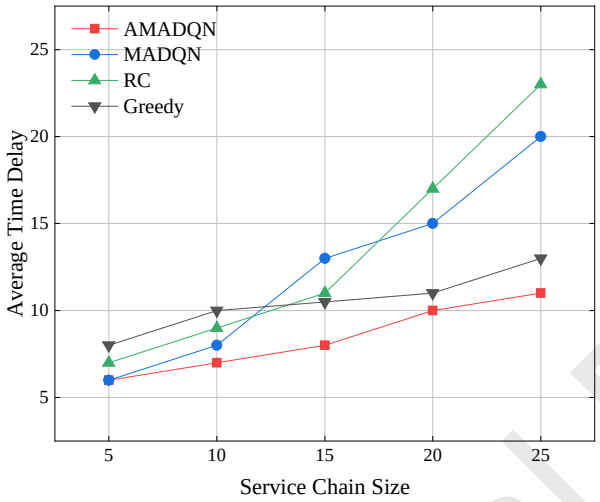


Fig. 7. Performance comparison of A-MADQN and other benchmark methods for varying service chain sizes.

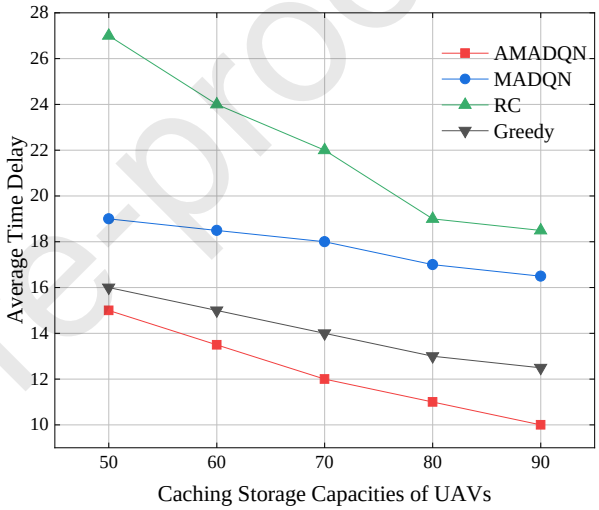


Fig. 9. Performance comparison of A-MADQN and other benchmark methods across varying caching storage capacities of UAVs.

ity regardless of UAV density. The reason is that additional multi-rotor UAVs enable the Greedy algorithm to cache more services, thereby be meeting vehicle user requests in a better way.

Fig. 7 compares the performance comparison of A-MADQN with other benchmark methods across various service chain sizes. As the service chain size increases, the average time delay for all methods rises, due to the greater number of tasks that need to be processed. Notably, A-MADQN consistently outperforms other methods across different service chain sizes.

Figs. 8 and 9 compare performance under varying UAV computing and caching storage capacities. As these capacities increase, the average time delay for all methods decreases. This improvement is due to higher computing capacities accelerating task processing and larger caching storage capacities minimizing frequent data updates. The A-MADQN algorithm consistently outperforms the other three benchmark methods, demonstrating its effectiveness in optimizing system performance across different resource conditions.

Fig. 10 illustrates the cache hit ratio of various schemes across different total service counts. As the total number of services in the environment increases, the CHR for all solutions gradually decreases. This decline occurs because, although initial resources may be sufficient, they become inadequate as the demand rises with the growing number of services. The A-MADQN algorithm outperforms other available solutions, while the RC algorithm falls short due to its lack of service caching consideration. Initially, when the total number of services is limited, the Greedy algorithm's performance is comparable to A-MADQN. However, as the total number of services increases, the performance gap between Greedy and A-MADQN widens. The Greedy approach focuses on the immediate, optimal choices. It reveals a critical weakness as service diversity grows. This short-sighted strategy becomes increasingly ineffective in environments with expanding service ranges, highlighting its limitations.

In summary, the proposed A-MADQN algorithm consistently demonstrates optimal performance across various scenarios, highlighting its effectiveness compared to other benchmark methods.

7. Conclusion

In this paper, we explores a two-tier UAV-assisted VECNs architecture where UAVs can selectively cache the previously generated service data for future reuse. To minimize the overall task chain delay, we examine the joint optimization of service chain caching and task offloading decisions, accounting for delay constraints. We propose a novel A-MADQN algorithm with a global reward. Extensive simulations demonstrate the high efficiency of our proposed algorithm. Future work could explore cache-switching penalties to address the frequent

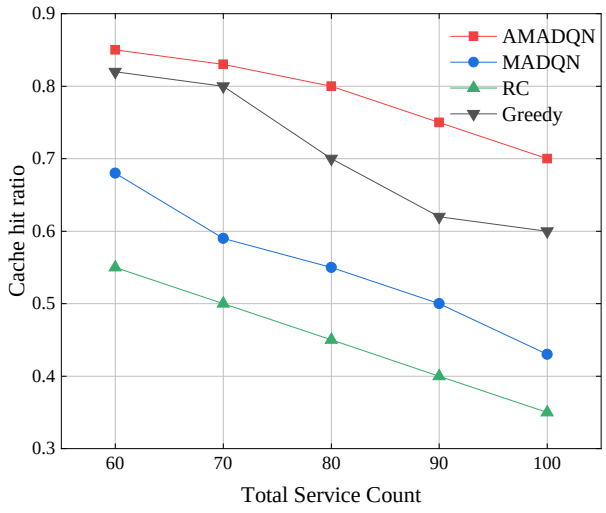


Fig. 10. Comparison of cache hit ratio for various total service counts.

updates caused by dynamic task arrivals, potentially leading to more efficient cache management strategies and improved overall system performance.

Acknowledgements

This work is supported by Guangdong Basic and Applied Basic Research Foundation (Grant No. 2024A1515012745)

References

[1] B. Li, Z. Fei, Y. Zhang, Uav communications for 5g and beyond: Recent advances and future trends, IEEE Internet of Things Journal. 6 (2) (2018) 2241–2263.

[2] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, M. Chen, In-edge ai: Intelligentizing mobile edge computing, caching and communication by federated learning, Ieee Network. 33 (5) (2019) 156–165.

[3] K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, S. Salloum, M. S. Mahmud, A review of optimization methods for computation offloading in edge computing networks, Digital Communications and Networks. 9 (2) (2022) 450–461.

[4] Y. Li, C. Liao, Y. Wang, C. Wang, Energy-efficient optimal relay selection in cooperative cellular networks based on double auction, IEEE Transactions on Wireless Communications. 14 (8) (2015) 4093–4104.

[5] S. Xia, Z. Yao, Y. Li, S. Mao, Online distributed offloading and computing resource management with energy harvesting for heterogeneous mec-enabled iot, IEEE Transactions on Wireless Communications. 20 (10) (2021) 6743–6757.

[6] T. X. Tran, D. Pompili, Joint task offloading and resource allocation for multi-server mobile-edge computing networks, IEEE Transactions on Vehicular Technology. 68 (1) (2018) 856–868.

[7] X. Zhong, Y. Guo, N. Li, Y. Chen, S. Li, Deployment optimization of uav relay for malfunctioning base station: Model-free approaches, IEEE Transactions on Vehicular Technology. 68 (12) (2019) 11971–11984.

[8] L. Zhou, H. Mao, X. Deng, J. Zhang, H. Zhao, J. Wei, Real-time radio map construction and distribution for uav-assisted mobile edge computing networks, IEEE Internet of Things Journal. 11 (12) (2024) 21337–21346.

[9] F. Zhou, Y. Wu, R. Q. Hu, Y. Qian, Computation rate maximization in uav-enabled wireless-powered mobile-edge computing systems, IEEE Journal on Selected Areas in Communications. 36 (9) (2018) 1927–1941.

[10] Y. Luo, W. Ding, B. Zhang, Optimization of task scheduling and dynamic service strategy for multi-uav-enabled mobile-edge computing system, IEEE Transactions on Cognitive Communications and Networking. 7 (3) (2021) 970–984.

[11] Y. Wang, Z.-Y. Ru, K. Wang, P.-Q. Huang, Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing, IEEE transactions on cybernetics. 50 (9) (2019) 3984–3997.

[12] H. Peng, X. Shen, Multi-agent reinforcement learning based resource management in mec-and uav-assisted vehicular networks, IEEE Journal on Selected Areas in Communications. 39 (1) (2020) 131–141.

[13] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, L. Hanzo, Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing, IEEE Transactions on Cognitive Communications and Networking. 7 (1) (2020) 73–84.

[14] Y. Zhou, C. Pan, P. L. Yeoh, K. Wang, M. Elkashlan, B. Vucetic, Y. Li, Secure communications for uav-enabled mobile edge computing systems, IEEE Transactions on Communications. 68 (1) (2019) 376–388.

[15] M. Abrar, U. Ajmal, Z. M. Almohaimeed, X. Gui, R. Akram, R. Masroor, Energy efficient uav-enabled mobile edge computing for iot devices: A review, IEEE Access. 9 (2021) 127779–127798.

[16] Y. Ding, Z. Yang, Q.-V. Pham, Y. Hu, Z. Zhang, M. Shikh-Bahaei, Distributed machine learning for uav swarms: Computing, sensing, and semantics, IEEE Internet of Things Journal. 11 (5) (2024) 7447–7473.

[17] Y. Li, H. Ma, L. Wang, S. Mao, G. Wang, Optimized content caching and user association for edge computing in densely deployed heterogeneous networks, IEEE Transactions on Mobile Computing. 21 (6) (2022) 2130–2142.

[18] J. Zheng, Q. Zhu, A. Jamalipour, Content delivery performance analysis of a cache-enabled uav base station assisted cellular network for metaverse users, IEEE Journal on Selected Areas in Communications. 42 (3) (2024) 643–657.

[19] M. Wu, K. Li, L. Qian, Y. Wu, I. Lee, Secure computation offloading and service caching in mobile edge computing networks, IEEE Communications Letters. 28 (2) (2024) 432–436.

[20] X. Cui, R. Hu, Application of intelligent edge computing technology for video surveillance in human movement recognition and taekwondo training, Alexandria Engineering Journal. 61 (4) (2022) 2899–2908.

[21] X. Chen, L. Jiao, W. Li, X. Fu, Efficient multi-user computation offloading for mobile-edge cloud computing, IEEE/ACM transactions on networking. 24 (5) (2015) 2795–2808.

[22] N. Abbas, Y. Zhang, A. Taherkordi, T. Skeie, Mobile edge computing: A survey, IEEE Internet of Things Journal. 5 (1) (2017) 450–465.

[23] M. Mozaffari, W. Saad, M. Bennis, M. Debbah, Drone small cells in the clouds: Design, deployment and performance analysis, in: 2015 IEEE global communications conference (GLOBECOM), IEEE, 2015, pp. 1–6.

[24] J. Chen, S. Chen, S. Luo, Q. Wang, B. Cao, X. Li, An intelligent task offloading algorithm (itoa) for uav edge computing network, Digital Communications and Networks. 6 (4) (2020) 433–443.

[25] Z. Yu, Y. Gong, S. Gong, Y. Guo, Joint task offloading and resource allocation in uav-enabled mobile edge computing, IEEE Internet of Things Journal. 7 (4) (2020) 3147–3159.

[26] E. El Haber, H. A. Alameddine, C. Assi, S. Sharafeddine, Uav-aided ultra-reliable low-latency computation offloading in future iot networks, IEEE Transactions on Communications. 69 (10) (2021) 6838–6851.

[27] B. Liu, W. Zhang, W. Chen, H. Huang, S. Guo, Online computation offloading and traffic routing for uav swarms in edge-cloud computing, IEEE Transactions on Vehicular Technology. 69 (8) (2020) 8777–8791.

[28] Y. Sun, S. Zhou, J. Xu, Emm: Energy-aware mobility management for mobile edge computing in ultra dense networks, IEEE Journal on Selected Areas in Communications. 35 (11) (2017) 2637–2646.

[29] Y. Mao, J. Zhang, K. B. Letaief, Joint task offloading scheduling and transmit power allocation for mobile-edge computing systems, in: 2017 IEEE wireless communications and networking conference (WCNC), IEEE, 2017, pp. 1–6.

[30] L. Huang, S. Bi, Y.-J. A. Zhang, Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks, IEEE Transactions on Mobile Computing. 19 (11) (2019) 2581–2593.

[31] M. Chen, Y. Hao, Task offloading for mobile edge computing in software defined ultra-dense network, IEEE Journal on Selected Areas in Communications. 36 (3) (2018) 587–597.

[32] J. Xu, L. Chen, P. Zhou, Joint service caching and task offloading for mobile edge computing in dense networks, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, IEEE, 2018, pp. 207–215.

[33] X. He, R. Jin, H. Dai, Joint service placement and resource allocation for multi-uav collaborative edge computing, in: 2021 IEEE Wireless Communications and Networking Conference (WCNC), IEEE, 2021, pp. 1–6.

[34] J. Yan, S. Bi, L. Duan, Y.-J. A. Zhang, Pricing-driven service caching and task offloading in mobile edge computing, IEEE Transactions on Wireless Communications. 20 (7) (2021) 4495–4512.

[35] S. Bi, L. Huang, Y.-J. A. Zhang, Joint optimization of service caching placement and computation offloading in mobile edge computing systems, IEEE Transactions on Wireless Communications. 19 (7) (2020) 4947–4963.

[36] G. Zhao, H. Xu, Y. Zhao, C. Qiao, L. Huang, Offloading dependent tasks in mobile edge computing with service caching, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 1997–2006.

[37] Y. Han, Z. Zhao, J. Mo, C. Shu, G. Min, Efficient task offloading with dependency guarantees in ultra-dense edge networks, in: 2019 IEEE Global Communications Conference (GLOBECOM), IEEE, 2019, pp. 1–6.

[38] B. Xu, Z. Kuang, J. Gao, L. Zhao, C. Wu, Joint offloading decision and trajectory design for uav-enabled edge computing with task dependency, IEEE Transactions on Wireless Communications. 22 (8) (2023) 5043–5055.

[39] C. Zhang, L. Zhang, L. Zhu, T. Zhang, Z. Xiao, X.-G. Xia, 3d deployment of multiple uav-mounted base stations for uav communications, IEEE Transactions on Communications. 69 (4) (2021) 2473–2488.