2023 IEEE 14th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON) | 979-8-3503-0413-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/UEMCON59035.2023.10316024

Semantic-Aware Blockchain Architecture Design for Edge-enabled Metaverse

Ning Wang, *Student Member, IEEE,* Beatriz Lorenzo, *Senior Member, IEEE*
Dept. Electrical and Computer Engineering, University of Massachusetts Amherst, USA
{ninwang, blorenzo}@umass.edu

***Abstract*—The evolution of the Metaverse places higher demands on task transmission and computation to provide users with enhanced quality services. In this paper, we are first to address the timely task allocation and processing for heterogeneous services described using semantics in the Metaverse. We consider a mobile edge computing network composed of Unmanned Aerial Vehicles (UAVs) and adopt blockchain technology to support secure task allocation. The task allocation problem is formulated using semantic matching and penalty mechanisms and solved by Particle Swarm Optimization (PSO) and Learning-based algorithms. We have analyzed the impact of the penalty mechanism on system rewards, task server matching rates, and the success rate of task service. Our results show significant improvements of our architecture and algorithms compared to existing approaches.**

***Index Terms*—Metaverse, semantic awareness, mobile edge computing (MEC), task allocation, blockchain, reinforcement learning.**

I. Introduction

In recent years, the Metaverse has exhibited its potential across diverse domains, e.g., military communications, smart cities, and eHealth [1]. Facebook's rebranding as Meta exemplifies its drive for virtual world engagement [2]. Essential Metaverse technologies encompass smart sensing, mobile edge computing (MEC), blockchain, and digital twins, as described in [3]. Wang et al. [4] present a semantic transmission framework for device-to-Metaverse data transfer. Semantic-aware communication, a task-oriented method, analyzes agent state and action during task execution, eliminating the need for extra data source processing [5].

MEC is proposed as a Metaverse architecture to bring computing resources closer to users, reduce network congestion, improve energy efficiency, and achieve real-time interaction [6], [7]. Yet, efficient MEC deployment requires good server density, location, capabilities, and mobility to meet heterogeneous demands. Zhang et al. [8] introduce a location-based MEC-driven Metaverse, wherein servers adapt augmented reality resolution to scarce communication and computing resources. Al-Shuwaili and Simeone[9] presents an energy-efficient resource allocation approach over communication and computation resources via successive convex approximation.

Preserving users' privacy during communication and computation processes is a challenge effectively resolved by blockchain, enabling dynamic network management, trust, and payments for diverse services. A blockchain-based, reinforcement learning-driven model was suggested in [10] for Metaverse task allocation on static servers, improving system gains. Lan et al. [11] proposed a blockchain-based UAV-assisted IoT data collection scheme, optimizing transmissions and deployment to maximize throughput. Particle Swarm Optimization (PSO) efficiently optimizes UAV positions in [12]. A dual-agent model to optimize UAV trajectory and channel allocation is introduced in [13] .

This work is partially supported by the US National Science Foundation under Grant CNS-2008309.

Despite existing works, the joint UAV placement, communication and computing resource allocation for semantic tasks in the Metaverse remains unexplored. Therefore, in this paper we propose a new Metaverse architecture utilizing a blockchain network composed of UAVs for computing different semantic tasks generated by Metaverse users, aiming to achieve maximum throughput and system revenue. Our approach significantly improves the system performance by efficiently utilizing computing resources, and thus increasing the operators' revenues.

The main contribution of this work can be summarized as following:

- We present a model that strategically optimizes the distribution of UAV servers to enhance system throughput. This model capitalizes on the Particle Swarm Optimization (PSO) algorithm's rapid convergence capabilities.
- We employ blockchain to construct a model that enables the sharing of user-generated task information, preserving location privacy, and concurrently facilitating server participation and profit distribution. Through the integration of Q-learning and semantic task matching with penalty mechanisms, we optimize both communication and computation processes, leading to improved resource utilization and overall returns.
- The simulation results show that our Metaverse architecture design and the PSO and Q-learning algorithms proposed have significant advantages compared with greedy and random selection algorithms.

The rest of the paper is organized as follows. Section II introduces our architecture, communication, and computing models. The task and resource allocation problem is formulated in Section III. Section IV describes our proposed algorithms. Simulation results are presented in Section V.

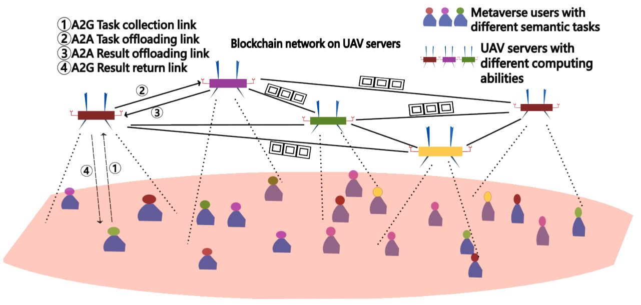

① A2G Task collection link
② A2A Task offloading link
③ A2A Result offloading link
④ A2G Result return link
Blockchain network on UAV servers
Metaverse users with different semantic tasks
UAV servers with different computing abilities
②
③
④
①

Fig. 1: UAV-enabled and semantic-aware blockchain Metaverse architecture.

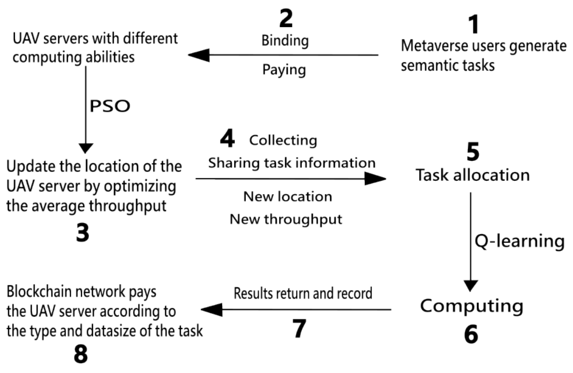

2
Binding
Paying
1
Metaverse users generate semantic tasks
UAV servers with different computing abilities
PSO
Update the location of the UAV server by optimizing the average throughput
3
4 Collecting
Sharing task information
New location
New throughput
5
Task allocation
Q-learning
Computing
6
Results return and record
7
Blockchain network pays the UAV server according to the type and datasize of the task
8

Fig. 2: System flowchart.

Section VI concludes the paper and summarizes the future work.

II. System model

A. System Overview

The proposed UAV-enabled and semantic-aware blockchain architecture for Metaverse is shown in Fig. 1. It includes Metaverse users with computational task requests, UAV servers, and a consortium blockchain system. Tasks generated by users in different locations are assigned serial numbers according to their collection time. The set of tasks is represented by $\Phi = \{\phi_1, \cdots, \phi_j, \cdots, \phi_m\}$, the task index is j, and the total number of the tasks is m. The set of UAV servers is denoted by $\mathcal{S} = \{s_1, \cdots, s_i, \cdots, s_n\}$ with i the index of the server, and n the total number of servers.

The workflow of this system is described in the following steps as illustrated in Fig. 2:

1) Task generation and semantic classification: We consider that Metaverse applications will generate tasks with different semantic information, translating into heterogeneous communication and computing requirements, and price. We use $k \in \{1, 2, \cdots, K\}$ to represent the semantic type. Similarly, servers are divided into K categories according to their ability to process semantic information.

2) User-UAV server binding: Initially, we bind each user to the closest UAV as the trusted server to collect data and upload it to the blockchain network for allocation of computing resources and return the results in a timely manner. We define tasks collected by server s_i as $\Phi_{s_i} = \{\phi_1, \cdots, \phi_{j_i}, \cdots, \phi_{m'}\}$, where $m' \leq m$ is the number of tasks collected by s_i.

3) UAV deployment: Many metaverse users generate computing tasks at the same time. Therefore, each UAV should be deployed in a location to optimize the average throughput of different communication channels. We denote the location of UAV servers by the set $Z = \{\zeta_1, \cdots, \zeta_i, \cdots, \zeta_n\}$. We use $\zeta_i\,(x_i, y_i)$ to represent the initial position and $\zeta_i'\,(x_i', y_i')$ to be the assigned locations obtained to achieve a balance between the throughput of the UAV servers for collecting and transmitting data within the blockchain, and improve the profitability of the blockchain system.

4) Data collection: UAV servers collect the users' data and their task descriptions. We assume each user generates one computing task at a time and shares the location with their trusted server. The description of task ϕ_j is represented by $I_j < k^j, req_{k^j}, p_{k^j}, D_j >$, where k^j is the semantic type, req_{k^j} is the user's task requirement based on the semantic type k^j given in terms of delay and throughput [14], p_{k^j} is the unit price per CPU cycle for processing the task ϕ_j, D_j is the task data size.

5) Task allocation: Once a UAV server s_i receives the data and I_j, the data is stored locally, and I_j is submitted to the blockchain network for task allocation to decide which server will process the data.

6) Offloading and computing: Based on the task allocation decision, server s_i will either forward the received task to the assigned UAV server for computing or compute it locally. Once the computing task is completed, each server will broadcast the results to the blockchain network.

7) Results return and reward payment: UAV servers will receive their own users' tasks computing results and send them back to their respective users. Then, they will get their reward payment through the blockchain based on the paid offloading fees by the users.

B. Communication Model

We use four transmission links to complete task collection, task offloading, result offloading, and result return, as illustrated in Fig. 1. The task collection and result return process (links 1 and 4 in Fig. 1) can be modeled as an air-to-ground (A2G) channel where line-of-sight (LoS) path loss and none-line-of-sight (NLoS) [15] are

$$PL_\xi = \left(\frac{4\pi f_c}{c}\right)^2 \cdot d_{i,j}^2 \cdot \eta_\xi \tag{1}$$

where $d_{i,j}$ is the distance between the UAV server s_i and the metaverse user who generates task ϕ_j, c is the speed of light, f_c is the carrier frequency, and η_ξ with $\xi = \{0, 1\}$ is the path loss of LoS and NLoS cases. The average A2G path loss for the task collection and result return transmissions between server s_i and user j can be obtained by

$$\overline{L_{i,j}} = p_0 \cdot PL_0^{i,j} + p_1 \cdot PL_1^{i,j} \tag{2}$$

where p_0 and p_1 are the probabilities of LoS and NLoS [15] respectively. Therefore, the transmission rate between server i and user j is given by

$$R_{i,j} = B_i log_2 \left(1 + \frac{P^i G_{i,j}}{\overline{L_{i,j}} N_0 B_i}\right) \tag{3}$$

where B_i is the bandwidth, $P^i = P^i_{col}$ represents the transmission power for data collection of server s_i and $P^i = P^i_{ret}$ is the transmission power to return the result, $G_{i,j}$ is the channel gain, $\overline{L_{i,j}}$ is the average path loss, and N_0 is the noise power spectral density. The task collection duration is $T^{col}_{i,j} = \frac{D_j}{R_{i,j}}$ for the task data size D_j, and the time of result return is $T^{ret}_{i,j} = \frac{D'_j}{R_{i,j}}$ with D'_j the data size of the result. Then, the energy consumption for data collection and return is obtained as $E^{col}_{i,j} = P^i_{col} T^{col}_{i,j}$ and $E^{ret}_{i,j} = P^i_{ret} T^{ret}_{i,j}$, respectively.

The air-to-air (A2A) link for the task and results offloading process between UAV servers can be modeled as a free space channel model. The path loss between servers s_i and $s_{i'}$ [16]

$$L_{i,i'} = PL_0 = \left(\frac{4\pi f_c}{c}\right)^2 \cdot d^2_{i,i'} \cdot \eta_0 \tag{4}$$

where η_ξ with $\xi = 0$ represents the path loss of LoS case, $d_{i,i'}$ is the distance between server s_i and $s_{i'}$. Thus, the transmission rate of the A2A link can be expressed as

$$R_{i,i'} = B_i log_2 \left(1 + \frac{P^i_{tr} G_{i,i'}}{L_{i,i'} N_0 B_i}\right) \tag{5}$$

where P^i_{tr} is the transmission power of server i and $G_{i,i'}$ is the channel gain. Then, the time needed to offload the task and the result is $T^{tof}_{i,i',j} = \frac{D_j}{R_{i,i'}}$ and $T^{rof}_{i,i',j} = \frac{D'_j}{R_{i,i'}}$, respectively. Similarly, the energy consumption for each case is $E^{tof}_{i,i',j} = P^i_{tr} T^{tof}_{i,i',j}$ and $E^{rof}_{i,i',j} = P^i_{tr} T^{rof}_{i,i',j}$.

C. Computing Cost Model

We have K types of UAV servers with different computing resources and abilities. The total CPU cycles required to compute task ϕ_j of data size D_j on the UAV server s_i is $\mu_{i,j} = D_j \theta_{k^i}$ with θ_{k^i} the CPU cycles required to process unit data sample. Let α_{k^i} represent the parameter related to the architecture of CPU for server s_i and f_{k^i} the CPU frequency to compute the offloading tasks. Then, the energy cost is calculated as $E^{comp}_{i,j} = \alpha_{k^i} \mu_{i,j} f^2_{k^i}$ [17]. Moreover, we can obtain the time consumption for computing task ϕ_j on the server s_i as $T^{comp}_{i,j} = \frac{\mu_{i,j}}{f_{k^i}}$, where μ_{k^i} is the maximum available computing capacity of server s_i.

D. Blockchain Network

The blockchain network runs on the UAV servers using the PTFT protocol [18] to aid in the transmission and preserve the privacy of information and results, all while guaranteeing a secure payment for services rendered. The blockchain network in our system operates as follows: 1) At the beginning, each UAV server publishes its location, available communication and computing resources for processing tasks, the types of semantic information it excels at processing, and uploads received task descriptions I_j to the blockchain network. 2) The previous information will be broadcast in the blockchain network to allocate the received tasks to appropriate UAV servers. Next, the task allocation decisions, task descriptions, and resource and location information of the UAV servers will be packaged into a new block. 3) If s_i is assigned to process its own received task, it will do local computing. Otherwise, it will transfer the task to other servers for processing. 4) Once the task is completed, the result is recorded on the blockchain and transmitted back to the server that sent the data. 5) The users receive results from the trusted UAV server, and the involved servers will be paid according to their participation.

III. Problem Formulation

A. Utility Definitions

We formulate the task allocation problem considering the different communication and computing costs when the task is computed locally (by the closest trusted UAV that collects the data) or offloaded to other UAVs. In addition, we also consider if the task is offloaded to a UAV that handles tasks of the same semantic type or different.

For the UAV development model, the utility of a UAV server s_i that flies to collect the tasks $\Phi_{s_i} = \{\phi_1, \phi_2, ..., \phi_{m'}\}$ from the corresponding users and communicates with other servers is

$$U^{dep}_{i,j,m'} = \beta \left(\sum_{j=1}^{m'} \left(\frac{R^{A2G}_{i,j}}{m'}\right)\right) + (1-\beta)\left(\sum_{i'=1}^{n}\left(\frac{R^{A2A}_{i,i'}}{n-1}\right)\right) \tag{6}$$

where $\beta\,(0 \le \beta \le 1)$ is the weighting factor between the A2G channel and A2A channel throughputs of UAV s_i, m' is the number of users binding with s_i, n is the number of UAV servers, and $n-1$ refers to other servers excluding server s_i. The average throughput of the A2G channel is $R^{A2G}_{i,j} = \frac{R^{A2G}_{col} + R^{A2G}_{ret}}{2}$, where $R^{A2G}_{col} = R_{i,j}(P^i_{col})$ and $R^{A2G}_{ret} = R_{i,j}(P^i_{ret})$ as in (3). $R^{A2A}_{i,i'}$ is the throughput of the A2A channel from server s_i to all other servers s'_i as in (5). Therefore, (6) represents each server's average throughput of the A2A and A2G links.

For the internal model of the blockchain, we use $\mathbf{1}_{i,j} = 1$ to indicate that task ϕ_j is assigned to server s_i for computing. Otherwise, $\mathbf{1}_{i,j} = 0$.

If task j is submitted by server i, $\phi_j \in \Phi_{s_i}$, then the utility can be expressed as:

$$\begin{aligned} U^{blo}_{i,j} = &\mathbf{1}_{i,j}\left(p_{k^j}\mu_{i,j} - E^{comp}_{i,j} - \rho_{k^i,k^j} p_{k^j}\mu_{i,j}\right) \\ &+ (1-\mathbf{1}_{i,j})\left(\lambda p_{k^j}\mu_{i,j} - E^{tof}_{i,i',j} - \nu_{k^i,k^j} p_{k^j}\mu_{i,j}\right) \end{aligned} \tag{7}$$

where $\mathbf{1}_{i,j} = 1$ means the task is assigned to be computed locally. In that case, the profit is the remuneration paid by the users minus the computing energy consumption and the penalties for mismatches proportional to the amount of data transmitted. ρ_{k^i,k^j} is a penalty coefficient for computing which equals 0 if $k^i = k^j$. $\mathbf{1}_{i,j} = 0$ means the task is not assigned to be computed locally, and the local UAV server will transmit it to a certain server, charging a certain intermediary service fee. In this case, the profit is the intermediary service fee minus the task offloading energy consumption and the penalty when the task is delegated to a server that does not match the semantic task type. λ is the proportion of intermediary fees. ν_{k^i,k^j} is a penalty coefficient for communication, and it equals 0 if $k^i = k^j$.

If task j is not submitted by server i, $\phi_j \notin \Phi_{s_i}$, the utility can be expressed as:

$$\begin{aligned} U_{i,j}^{blo} = \mathbf{1}_{i,j} \big(& (1-\lambda) p_{k^j} \mu_{i,j} - \rho_{k^i,k^j} p_{k^j} \mu_{i,j} \\ & - E_{i,j}^{comp} - E_{i,i',j}^{rof} \big) \end{aligned} \tag{8}$$

where $\mathbf{1}_{i,j} = 1$ indicates that the task is assigned to be computed in a certain UAV server and not locally. The profit consists of the remuneration minus the communication intermediary fee, the energy consumption caused by calculation and task return, and the penalties for semantic mismatches. The rest of the parameters are described above. $\mathbf{1}_{i,j} = 0$ means this server is not involved in the communication and calculation process, and thus it will not receive any profit.

B. UAV Deployment, Task, and Resource Allocation

First, we aim to optimize the position to deploy each UAV server ζ_i' to maximize the overall average throughput for A2G and A2A channels. The distance traveled by the UAV to collect the data is denoted by χ_i. This optimization problem is formulated as:

$$\begin{aligned} (P1): \quad & \arg\max_{\zeta_i'} \sum_{i=1}^{n} U_{i,j,m'}^{dep} \\ \text{s.t.}: \quad & \text{a)}: 0 \le m' \le m, \\ & \text{b)}: \chi_i \ge 0, \quad \forall i, \\ & \text{c)}: i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m'\}. \end{aligned} \tag{9}$$

where $U_{i,j,m'}^{dep}$ is given by (6), (9a) constraints the number of users that a UAV can serve in a period of time; (9b) guarantees each UAV server's movement is reasonable; (9c) defines the set of servers and tasks collected.

Then, we optimize the task allocations to UAV servers in the blockchain with semantic awareness to maximize the profit. The problem is formulated as

$$\begin{aligned} (P2): \quad & \arg\max_{\mathbf{1}_{i,j}} \sum_{i=1}^{n} \sum_{j=1}^{m} U_{i,j}^{blo} \\ \text{s.t.}: \quad & \text{a)}: \sum_{i=1}^{n} \mathbf{1}_{i,j} \le 1, \quad \forall j, \\ & \text{b)}: \sum_{j=1,k^j=k^i}^{m} \mu_{i,j} \mathbf{1}_{i,j} + \sum_{j=1,k^j \ne k^i}^{m} \mu_{i,j} \mathbf{1}_{i,j} \le \mu_{k^i}, \\ & \text{c)}: \sum_{j=1,k^j \ne k^i}^{m} \mu_{i,j} \mathbf{1}_{i,j} \le \delta \mu_{k^i}, \quad 0 \le \delta \le 1, \\ & \text{d)}: \zeta_i' \quad \text{is constant}, \quad \forall i, \\ & \text{e)}: T_{i,j}^{task} \le req_{k^j}^{T}, \quad \forall \mathbf{1}_{i,j} = 1, \\ & \text{f)}: R_{i,i'} \ge req_{k^j}^{R}, \quad \forall \phi_j \in \Phi_{s_i} \ \& \ \mathbf{1}_{i',j} = 1, \\ & \text{g)}: i \in \{1,2,\cdots,n\}, j \in \{1,2,\cdots,m\}. \end{aligned} \tag{10}$$

where $U_{i,j}^{blo}$ is defined in (7)-(8), (10a) guarantees that one task can only be assigned to one server; (10b) is the computing capacity constraint to ensure that tasks served by s_i do not exceed its computing resources; (10c) limits the number of semantic tasks allocated to a different type of server by a ratio δ to ensure that servers can reserve enough computing resources to serve tasks of their same type; (10d) ensures that after each UAV is deployed, the location will not change when the blockchain is running; (10e) constrains the time that the task ϕ_j remains in the blockchain network to meet the time limit for the task of type k where $T_{i,j}^{task} = T_{i,i',j}^{tof} + T_{i',j}^{comp}$; (10f) indicates that the selected server for offloading the task must achieve a throughput $R_{i,i'}$ that meets the required one by the tasks semantic type; and (10g) defines the set of servers and tasks collected, respectively.

IV. PSO and Learning Based Solution

To cope with the complexity of solving the previous optimization problems, we adopt PSO algorithm [12] to find optimal positions to deploy UAVs due to its fast convergence and robustness, and reinforcement learning with ϵ-greedy strategy to perform the Q-learning algorithm for task allocation. Both methods require one UAV agent to assist with the execution.

A. Problem 1 Reformulation

To solve problem 1 with PSO algorithm, we define the state space, rules for updating, and reward function.

1) State space: We describe the state space as $\Omega_1 =< \mathcal{S}, \mathcal{Z}, \Phi' >$ that includes the set of servers $\mathcal{S}$, the set of their initial locations $\mathcal{Z}$, and $\Phi' = \{\Phi_{s_1}, \cdots, \Phi_{s_i}, \cdots, \Phi_{s_n}\}$ is the set of the binding tasks to servers.

2) Updating rules: The PSO algorithm works based on swarm intelligence with a set of particles initially located at random positions and then adjusts their positions based on two learning parameters p_{best} and g_{best}. The first parameter is the best position of a single particle, while the second one is the best position of the particle swarm. Let $v_i\,(ite)$ denote the velocity of particle i at the iteration ite. The x and y

coordinates of particle i are updated according to the following formula:

$$v_i^x(ite+1) = \omega v_i^x(ite) + c_1 r_1 \left(p_{best}^{x_i}(ite) - x_i(ite)\right) + c_2 r_2 \left(g_{best}^{x_i}(ite) - x_i(ite)\right) \quad (11)$$

$$v_i^y(ite+1) = \omega v_i^y(ite) + c_1 r_1 \left(p_{best}^{y_i}(ite) - y_i(ite)\right) + c_2 r_2 \left(g_{best}^{y_i}(ite) - y_i(ite)\right) \quad (12)$$

$$x_i(ite+1) = x_i(ite) + v_i^x(ite+1) \quad (13)$$

$$y_i(ite+1) = y_i(ite) + v_i^y(ite+1) \quad (14)$$

where ω is the inertia coefficient, c_1, c_2 are respectively the personal and global acceleration coefficients, r_1 and r_2 are two random values in the range of $(0,1)$.

3) Reward Function: The objective of problem 1 is to find the best location for each UAV server to optimize the overall average throughput for UAV servers. Therefore, the reward function is

$$r_{i,j,m'}^{dep} = U_{i,j,m'}^{dep} \quad (15)$$

where $U_{i,j,m'}^{dep}$ is as in (6) and we will find the best ζ_i' to maximize the reward.

B. Problem 2 Reformulation

We use the Q-learning algorithm to solve problem 2, and we reformulate the problem using a Markov Decision Process (MDP) with the state space, action space, and reward function as following.

1) State space: We describe the state space of the agent using $\Omega_2 =< \Phi, j_{type}, P, D, req^T, req^R >$, which comprises all tasks and their corresponding information. Here, $\Phi = \{\phi_1, \cdots, \phi_j, \cdots, \phi_m\}$ represents the set of tasks, $j_{type} = \{k^1, \cdots, k^j, \cdots, k^m\}$ denotes the set of semantic categories associated with each task. Each k^j is an integer between 1 and K, where K is the total number of semantic categories. $P = \{p_{k^1}, \cdots, p_{k^j}, \cdots, p_{k^m}\}$ represents the set of unit CPU cycle prices for tasks. $D = \{D_1, \cdots, D_j, \cdots, D_m\}$ corresponds to the set of data sizes of tasks. $req^T = \{req_{k^1}^T, \cdots, req_{k^j}^T, \cdots, req_{k^m}^T\}$ is the set of time requirements for different semantic tasks, and $req^R = \{req_{k^1}^R, \cdots, req_{k^j}^R, \cdots, req_{k^m}^R\}$ represents their throughput requirements.

2) Action space: We use $A_2 =< \mathcal{S}, i_{type}, M, \alpha, F, \Theta, B, P_{tr}, Z' >$ to represent the action space, where $\mathcal{S}$ is the set of servers, $i_{type} = \{k^1, \cdots, k^i, \cdots, k^n\}$ is the set of semantic categories corresponding to each server, where k^i is an integer between 1 and K, and K is the total number of semantic categories, $M = \{\mu_{k^1}, \cdots, \mu_{k^i}, \cdots, \mu_{k^n}\}$ is the set of total available CPU computing resources for each server, $F = \{f_{k^1}, \cdots, f_{k^i}, \cdots, f_{k^n}\}$ is the set of CPU frequencies, $\Theta = \{\theta_{k^1}, \cdots, \theta_{k^i}, \cdots, \theta_{k^n}\}$, $B = \{B_1, \cdots, B_i, \cdots, B_n\}$ is the set of the communication bandwidth, $P_{tr} = \{P_1^{tr}, \cdots, P_i^{tr}, \cdots, P_n^{tr}\}$ is the set of transmission power, $Z = \{\zeta_1, \cdots, \zeta_i, \cdots, \zeta_n\}$ is the set of each UAV server's location. The agent can choose only one action for per task, i.e., selected or not selected in one state, while one action can be selected several times in all states if it satisfies the necessary conditions. Through this setting, we ensure that a task can only be assigned to one server for calculation, and a server can calculate multiple tasks if the conditions are met.

3) Reward Function: The objective of problem 2 is to optimize the profit of the blockchain network and the time used for task processing by allocating semantic tasks to different UAV servers. Following the utility model, the reward function is defined as:

If $\phi_j \in \Phi_{s_i}$ and $k^j = k^i$, the reward function can be expressed as:

$$r_{i,j}^{blo} = \begin{cases} p_{k^j}\mu_{i,j} - E_{i,j}^{comp}, & \sum_{j=1}^m \mu_{i,j} \le \mu_{k^i}, T_{i,j}^{task} \le req_{k^j}^T, \\ & \sum_{j=1,k^j \ne k^i}^m \mu_{i,j} \le \delta\mu_{k^i}, \\ \lambda p_{k^j}\mu_{i,j} - E_{i,i',j}^{tof}, & Otherwise \;\; and \;\; k^{i'} = k^j, \\ \lambda p_{k^j}\mu_{i,j} - E_{i,i',j}^{tof} - \nu p_{k^j}\mu_{i,j}, & \\ & Otherwise \;\; and \;\; k^{i'} \ne k^j, \end{cases} \quad (16)$$

If $\phi_j \in \Phi_{s_i}$ and $k^j \ne k^i$, the reward function can be expressed as:

$$r_{i,j}^{blo} = \begin{cases} p_{k^j}\mu_{i,j} - E_{i,j}^{comp} - \rho p_{k^j}\mu_{i,j}, & \sum_{j=1}^m \mu_{i,j} \le \mu_{k^i}, \\ & T_{i,j}^{task} \le req_{k^j}^T, \sum_{j=1,k^j \ne k^i}^m \mu_{i,j} \le \delta\mu_{k^i}, \\ \lambda p_{k^j}\mu_{i,j} - E_{i,i',j}^{tof}, & Otherwise \;\; and \;\; k^{i'} = k^j, \\ \lambda p_{k^j}\mu_{i,j} - E_{i,i',j}^{tof} - \nu p_{k^j}\mu_{i,j}, & \\ & Otherwise \;\; and \;\; k^{i'} \ne k^j, \end{cases} \quad (17)$$

If $\phi_j \notin \Phi_{s_i}$, $\phi_j \in \Phi_{s_{i'}}$, $k^j = k^i$, the utility can be expressed as:

$$r_{i,j}^{blo} = \begin{cases} (1-\lambda)\, p_{k^j}\mu_{i,j} - E_{i,j}^{comp} - E_{i,i',j}^{rof}, & \\ & \sum_{j=1}^m \mu_{i,j} \le \mu_{k^i}, T_{i,j}^{task} \le req_{k^j}^T, \\ & \sum_{j=1,k^j \ne k^i}^m \mu_{i,j} \le \delta\mu_{k^i}, R_{i',i} \ge req_{k^j}^R, \\ 0, & Otherwise, \end{cases} \quad (18)$$

While if $\phi_j \notin \Phi_{s_i}$, $\phi_j \in \Phi_{s_{i'}}$, $k^j \ne k^i$, the utility can be expressed as:

$$r_{i,j}^{blo} = \begin{cases} (1-\lambda)\, p_{k^j}\mu_{i,j} - E_{i,j}^{comp} - E_{i,i',j}^{rof} - \rho p_{k^j}\mu_{i,j}, & \\ & \sum_{j=1}^m \mu_{i,j} \le \mu_{k^i}, T_{i,j}^{task} \le req_{k^j}^T, \\ & \sum_{j=1,k^j \ne k^i}^m \mu_{i,j} \le \delta\mu_{k^i}, R_{i',i} \ge req_{k^j}^R, \\ 0, & Otherwise, \end{cases} \quad (19)$$

These reward functions are the transformation of the utility and constraints to model various offloading decisions.

C. Implementation of PSO algorithm

The PSO algorithm to solve the reformulated problem 1 as described in Section IV.A is summarized in Algorithm 1.

Initially, particles are randomly generated around the map. Then each particle will calculate its own average throughput

Algorithm 1 PSO based algorithm to optimize locations of UAV servers

```
1: Initialize: PSO population, $g_{best}$, $p_{best}$, server_num,
   binding_users, location of other servers
2: for i ← 1 $to$ iteration do
3:     for each particle in the PSO population do
4:         Calculate the reward using (15) based on the current
           position of the particle
5:         Compare and update the personal best position $p_{best}$
6:         Compare and update the global best position $g_{best}$
7:         Update the particle's velocity $v_i^x$ and $v_i^y$ by (11) and
           (12)
8:         Update the particle's position $x_i$ and $y_i$ by (13) and
           (14)
9:     end for
10: end for
11: Output: The global best position $g_{best}$ after iteration
```

by (15). The best reward for each particle is p_{best} and for the swarm is the g_{best}, both of which will be updated and recorded. Then, we use (11) and (12) to update the velocity for each particle, and finally update the position x_i and y_i by using (13) and (14). This process will continue until the iteration reaches its maximum and ensure the convergence of the algorithm.

D. Implementation of Learning Algorithm

In problem 2, we use Q-learning to learn the task allocation problem. Before performing this learning process, we select the available UAV servers every time we allocate tasks. Here we use $S_j^{ava} = \{s_1, s_2, \cdots, s_n'\}$ to represent the available UAV servers to process task ϕ_j and $rewards_j = \{r_{1,j}, r_{2,j}, \cdots, r_{n,j}\}$ to represent the set of all possible rewards. h_j is the source server of task ϕ_j. The action is denoted by Act_j at state Ω_j. Ns means no service can be provided and episodes is the number of repeated experiments are conducted.

Here, we will elaborate on the Q-value updating method. For local processing and forwarding processing, we have employed two different strategies. When a server forwards a task, it considers not only the computing server's reward but also the reward of the forwarding source server. By doing so, we can introduce the calculation and communication penalties when tasks do not match servers and their impact on this system. As the output of the learning process we obtain the maximum reward of this blockchain network and the task allocation with the matching rate that indicates how many tasks are served by the same type of servers, the service rate is to see how many tasks achieve their strict requirements in this system. Otherwise, the system will not get any reward from the user.

V. Simulation Results

We have conducted extensive simulations to show the performance of our schemes. We set the map size to 200m*200m, $\lambda = 0.1$, the data size of each task is randomly generated between 20 and 30 Mbits. We have $K = 4$ semantic types

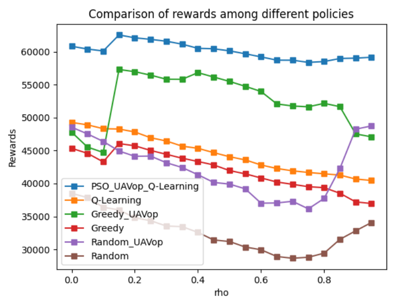
Algorithm 2 Learning based algorithm for task allocation

```
1: Input: $\phi_j$, $S$, $h_j$
2: Initialize: $rewards_j$, $S_j^{ava}$, $Act_j$, $Ns$,
       $\Omega_j$, episodes, $Q(\Omega_j, Act_j)$
3: for ep ← 1 to episodes do
4:     for j ← 1 to m do
5:         for i ← 1 to n do
6:             Using (16), (17), (18) and (19) to caculate reward
               $r_{i,j}$
7:             if $r_{i,j} > 0$ then
8:                 Append i into $S_j^{ava}$
9:             end if
10:        end for
11:        if length($S_j^{ava}$) == 0 then
12:            Append $\phi_j$ into $Ns$
13:        end if
14:        Generate a random value $x \in (0, 1)$
15:        if $x \le \epsilon$ then
16:            $Act_j$ ← randomly choose one from $S_j^{ava}$
17:        else
18:            $Act_j$ ← action with the max Q value from $S_j^{ava}$
19:        end if
20:        if $Act_j$ == $h_j$ then
21:            $Q(\Omega_j, Act_j) \leftarrow Q(\Omega_j, Act_j) + \alpha[r_{Act_j,j} + \gamma \cdot$
               $maxQ(\Omega_j', S_j^{ava}) - Q(\Omega_j, Act_j)]$
22:        else
23:            $Q(\Omega_j, Act_j) \leftarrow Q(\Omega_j, Act_j) + \alpha[r_{Act_j,j} + r_{h,j} +$
               $\gamma \cdot maxQ(\Omega_j', S_j^{ava}) - Q(\Omega_j, Act_j)]$
24:        end if
25:    end for
26:    Total_reward ← sum of the rewards; Matching_rate
       ← number of tasks matched with servers/m; Service_rate ← $(m - Ns)/m$.
27: end for
28: Max_reward: maximum total reward for all episodes
29: Allocation_result: allocation result for maximum rewards
30: Matching_rate: matching rate for maximum rewards
31: Service_rate: service rate for maximum rewards
32: Output: MAX_rewards, Allocation_result,
        Matching_rate, Service_rate
```

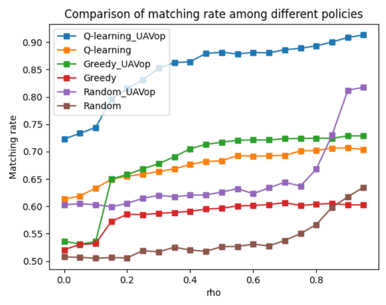
of tasks. The numbers of tasks and servers are 60 and 20, respectively. The available bandwidth for each type of task is 5, 25, 50, 100MHz, and the computing resources are 120, 100, 80, 60 CPU cycles/bit. The transmission power of UAV servers is randomly selected between 0.3 and 0.5W. The collection power is randomly selected between 0.05 and 0.1W. The antenna gain between the user and server is set to 20 dB. The carrier frequency is 2.5GHz, and the noise power spectral density is -130dB. The remaining parameters are described in Table I. Different semantic tasks' time and throughput requirements are [40ms, 1Mbit/s; 10ms, 5Mbit/s; 60ms, 10Mbit/s; 15ms, 20Mbit/s]. The results are the average of 100 independent simulations.

TABLE I: Simulation settings

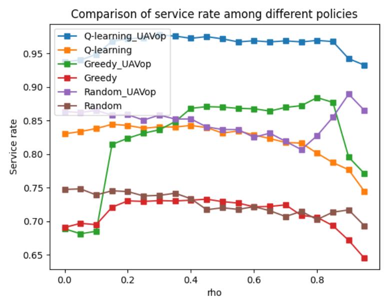
Parameter	**Description**	**Values**
p_{kj}	Unit price of computing for each type of task.	$[5, 10, 15, 20]$
θ_{ki}	CPU cycles required to process unit data.	$[2, 4, 6, 8]$
f_{ki}	CPU frequency used to compute task.	$[3, 6, 9, 12]$
α_{ki}	CPU architecture parameter.	$[0.01, 0.02, 0.03, 0.04]$
$a, b, \eta_{Los}, \eta_{NLos}$	Coefficients of A2G path loss.	$9.61, 0.16, 1, 20$
ϵ	The ratio of greedy strategy.	0.9
α	The learning rate.	0.5
γ	The discount factor.	0.9
$Episodes$	Max episode of P2.	500
ρ	Penalty coefficient for mismatched computing servers.	$[0, 0.9]$
ν	Penalty coefficient for mismatched forwarding servers.	$[0, 0.09]$
δ	Computing resource ratio for the unmatched task.	0.3
P_s	Population Size(Swarm Size).	100
w	Inertia Weight.	1
w_{damp}	Inertia Weight Damping Ratio.	0.98
c_1	Personal Learning Coefficient.	1.5
c_2	Global Learning Coefficient.	1.5
β	The weight factor of throughput.	0.6
$IterMax$	Max iteration of P1.	100



(a) Rewards vs. ρ

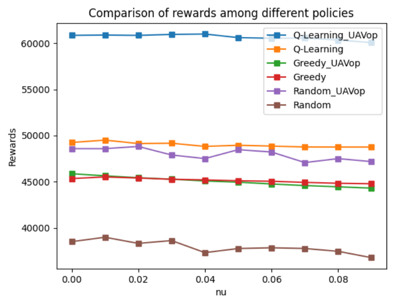


(b) Matching rate vs. ρ

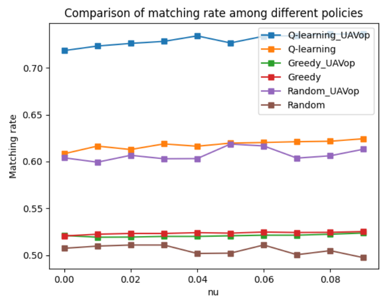


(c) Service rate vs. ρ

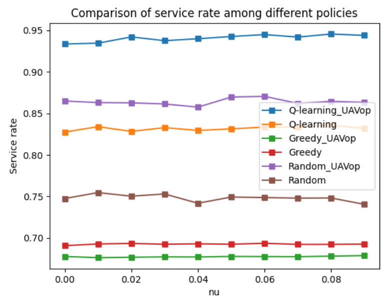
Fig. 3: Computational Penalty Mechanism Impacts on the System



(a) Rewards vs. ν



(b) Matching rate vs. ν



(c) Service rate vs. ν

Fig. 4: Communication Penalty Mechanism Impacts on the System

We analyze the impact of computational and communication penalties on the system's rewards, task-server matching rates, and task servicing success rate. We compare the performance of the Q-learning algorithm optimized with PSO for UAV positioning to the Q-learning algorithm without UAV optimization, as well as greedy and random selection algorithms.

As we can see in Fig. 3 and 4, the Q-learning algorithm optimized with PSO achieves the best performance by improving average throughput and increasing matching and service rates, resulting in higher system rewards. Optimizing UAV positions enables reliable task handling by selecting higher-revenue servers.

In terms of the computing penalties shown in Fig. 3, setting an appropriate penalty factor yields optimal system rewards around $\rho = 0.15$. This mechanism balances mismatched between task semantic type and server type, allowing partial mismatches that still contribute to overall rewards. However, increasing computation penalties gradually diminishes rewards as more mismatched allocations are replaced by matched ones. Under high computing penalties, the random selection algorithm experiences an increase in matching rate and overall rewards. This is because tasks initially assigned randomly to mismatched servers are forced to be reassigned to matching ones, improving rewards with higher matching rates. Appropriate computation penalties improve the number of successfully served tasks by reducing competition among demanding tasks for limited computing resources. Excessive penalties, however, decrease the system's service rate as high-performance mismatched servers lose opportunities for service.

Communication penalties, as shown in Fig. 4, have a smaller impact on the system compared to computation penalties due to the lower proportion of rewards derived from the communication process. The performance trends of the algorithms align with the previous analysis. The Q-learning algorithm optimized with PSO for UAV positioning achieves the best performance, finding an optimal point at $\nu = 0.04$. Although overall rewards remain relatively unchanged, a slight improvement in matching and service rates is observed with increasing communication penalty factor.

VI. Conclusion and Future work

In this paper, we have designed and implemented a UAV-enabled semantic-aware blockchain architecture for Metaverse applications. We solve the task allocation problem to maximize the throughput and revenue using the Q-learning algorithm with semantic matching and penalty mechanism. Moreover, we optimize UAV positioning using the Particle Swarm Optimization (PSO) algorithm. Our approach significantly outperformed existing schemes in terms of rewards, semantic matching rate, and service rate. In our future work, we will incorporate the optimization of computing resource allocation to servers to meet requirements of multiple coexisting Metaverse applications.

References

[1] M. Xu, W. C. Ng, W. Y. B. Lim, J. Kang, Z. Xiong, D. Niyato, Q. Yang, X. Shen, and C. Miao, "A full dive into realizing the edge-enabled metaverse: Visions, enabling technologies, and challenges," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, pp. 656–700, 2023.

[2] Y. Wang, Z. Su, N. Zhang, R. Xing, D. Liu, T. H. Luan, and X. Shen, "A survey on metaverse: Fundamentals, security, and privacy," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 319–352, 2023.

[3] F. Tang, X. Chen, M. Zhao, and N. Kato, "The roadmap of communication and networking in 6g for the metaverse," *IEEE Wireless Communications*, pp. 1–15, 2022.

[4] J. Wang, H. Du, Z. Tian, D. Niyato, J. Kang, and X. Shen, "Semantic-aware sensing information transmission for metaverse: A contest theoretic approach," *IEEE Transactions on Wireless Communications*, pp. 1–1, 2023.

[5] W. Yang, H. Du, Z. Q. Liew, W. Y. B. Lim, Z. Xiong, D. Niyato, X. Chi, X. Shen, and C. Miao, "Semantic communications for future internet: Fundamentals, applications, and challenges," *IEEE Communications Surveys Tutorials*, vol. 25, no. 1, pp. 213–250, 2023.

[6] K. Li, Y. Cui, W. Li, T. Lv, X. Yuan, S. Li, W. Ni, M. Simsek, and F. Dressler, "When internet of things meets metaverse: Convergence of physical and cyber worlds," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4148–4173, 2023.

[7] B. Lorenzo, J. Garcia-Rois, X. Li, J. Gonzalez-Castano, and Y. Fang, "A robust dynamic edge network architecture for the internet of things," *IEEE Network*, vol. 32, pp. 8–15, Jan 2018.

[8] H. Zhang, S. Mao, D. Niyato, and Z. Han, "Location-dependent augmented reality services in wireless edge-enabled metaverse systems," *IEEE Open Journal of the Communications Society*, vol. 4, pp. 171–183, 2023.

[9] A. Al-Shuwaili and O. Simeone, "Energy-efficient resource allocation for mobile edge computing-based augmented reality applications," *IEEE Wireless Communications Letters*, vol. 6, no. 3, pp. 398–401, 2017.

[10] Z. Wang, Q. Hut, M. Xu, and H. Jiang, "Blockchain-based edge resource sharing for metaverse," in *2022 IEEE 19th International Conference on Mobile Ad Hoc and Smart Systems (MASS)*, pp. 620–626, 2022.

[11] X. Lan, X. Tang, D. Zhai, D. Wang, and Z. Han, "Blockchain-secured data collection for uav-assisted iot: A ddpg approach," in *2021 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2021.

[12] N. Duong Thi Thuy, D. Nam Bui, M. Duong Phung, and H. Pham Duy, "Deployment of uavs for optimal multihop ad-hoc networks using particle swarm optimization and behavior-based control," in *2022 11th International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 304–309, 2022.

[13] P. Si, W. Yu, J. Zhao, K.-Y. Lam, and Q. Yang, "Uav aided metaverse over wireless communications: A reinforcement learning approach," 2023.

[14] H. Alves, G. D. Jo, J. Shin, C. Yeh, N. H. Mahmood, C. Lima, C. Yoon, N. Rahatheva, O.-S. Park, S. Kim, E. Kim, V. Niemelä, H. W. Lee, A. Pouttu, H. K. Chung, and M. Latva-aho, "Beyond 5g urllc evolution: New service modes and practical considerations," 2021.

[15] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multiagent actor–critic learning for age sensitive mobile-edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 1053–1067, 2022.

[16] S. Zhang, H. Zhang, B. Di, and L. Song, "Cellular uav-to-x communications: Design and optimization for multi-uav networks," *IEEE Transactions on Wireless Communications*, vol. 18, no. 2, pp. 1346–1359, 2019.

[17] Z. Wang, Q. Hu, R. Li, M. Xu, and Z. Xiong, "Incentive mechanism design for joint resource allocation in blockchain-based federated learning," 2022.

[18] M. Castro, B. Liskov, *et al.*, "Practical byzantine fault tolerance," in *OsDI*, vol. 99, pp. 173–186, 1999.