Graph Partition and Multiple Choice-UCB Based Algorithms for Edge Server Placement in MEC Environment

Zhao, Zheyu; Cheng, Hao; Xu, Xiaohua; Pan, Yi

DOI
10.1109/TMC.2023.3284994

Publication date
2023

Document Version
Final published version

Published in
IEEE Transactions on Mobile Computing

Citation (APA)
Zhao, Z., Cheng, H., Xu, X., & Pan, Y. (2023). Graph Partition and Multiple Choice-UCB Based Algorithms for Edge Server Placement in MEC Environment. *IEEE Transactions on Mobile Computing*, *23*(5), 4050-4061. https://doi.org/10.1109/TMC.2023.3284994

Graph Partition and Multiple Choice-UCB Based Algorithms for Edge Server Placement in MEC Environment

Zheyu Zhao, *Graduate Student Member, IEEE*, Hao Cheng, Xiaohua Xu, and Yi Pan, *Senior Member, IEEE*

***Abstract*— The deployment of edge servers make a significant impact on the service quality of a Mobile Edge Computing (MEC) system. This service quality relies on solving two key sub-problems: 1) interference management between servers 2) the placement of MEC servers. To improve the Quality of Service (QoS), we propose a method based on Graph Partition (GP) and Upper Confidence Bound (UCB) for solving these two sub-problems. Regarding interference management, we use an undirected graph to represent the interference between MEC servers so that the overall graph can be divided into multiple subsets of non-interfering MEC servers. Regarding server placement, we propose a Multiple Choice-Upper Confidence Bound (MC-UCB) algorithm that place an collection of interference aware edge servers in each selection. To evaluate the performance, we define a user's QoS function based on transmission delay, throughput, and user density comprehensively and compared with Particle Swarm Optimization (PSO) and Genetic Algorithm (GA) from previous work. The simulation results show that the performance of the proposed algorithms is improved by more than 4% compared with the GA algorithm and 6% compared with the PSO algorithm.**

***Index Terms*—Mobile edge computing, multiple choice-upper confidence bound, graph partition, edge server placement, interference management.**

I. Introduction

With the exponential growth of smart devices and the emergence of numerous new applications, network traffic has undergone a significant surge. However, the traditional centralized network structure struggles to meet the demands of users due to heavy data backhaul and long delays [1], [2]. To address these challenges, MEC has emerged as a distributed network architecture that leverages mobile base stations to extend cloud computing services to the network edge [3], [4]. MEC offers the advantage of breaking the physical limitations of mobile devices while alleviating the burden on cloud computing resources. Its strategic positioning in proximity to mobile users enables MEC to provide real-time access to wireless networks, high bandwidth, high throughput, and low latency, thus further enhancing the service experience for end-users [5].

There are two fundamental challenges that concerning user experience in MEC environment, one is interference management and the other is MEC server placement. In this article, interference management means the avoiding collisions of communications when multiple MEC servers broadcast simultaneously. When multiple users are located within the overlapping coverage areas of several MEC servers, the communication links requesting resources from different MEC servers can lead to interference, thereby reducing the overall channel quality. Notably, the probability of channel interference is particularly high between two neighboring MEC servers.

The strategy of the MEC server placement greatly affects the success rate of interference management. Ideally, deploying more edge computing servers can increase coverage and alleviate server resource competition. However, this approach leads to higher hardware deployment and operational costs. Therefore, it is crucial to strategically place a limited number of MEC servers to optimize costs in the MEC server placement problem. In this article, we aim to improve the user experience in the optimal way in terms of the servers quality. The first objective is to address the issue of interference management to optimize the quality of service for system users and the second objective is to manage the placement of these limited MEC servers. However, the issues of interference management and MEC server placement face many challenges.

1) First, many factors have an impact on system performance such as the number of MEC servers, hardware configuration, geographic distribution, user distribution, and channel status in the current scenario. The complexity of the problem grows linearly when the number of users and the number of MEC servers in the scenario increase.
2) Second, existing work mostly use heuristic algorithms to solve the problem of MEC server placement. However, in complex MEC scenarios, traditional heuristic algorithms encounter challenges such as slow search speed, high time

Manuscript received 3 March 2022; revised 31 May 2023; accepted 6 June 2023. Date of publication 21 June 2023; date of current version 4 April 2024. This work was supported in part by the National Key Research and Development Program of China under Grant 2021ZD0110403, in part by the National Natural Science Foundation of China (NSFC) under Grants 62172383 and 62231015, and in part by Anhui Provincial Key R&D Program under Grant S202103a05020098 Recommended for acceptance by D. T. Hoang. *(Corresponding author: Xiaohua Xu.)*

Zheyu Zhao and Xiaohua Xu are with the School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230026, China (e-mail: neuq_zhaozheyu@163.com; xiaohuaxu@ustc.edu.cn).

Hao Cheng is with the Department of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2628, CD Delft, The Netherlands (e-mail: calmch@hotmail.com).

Yi Pan is with the Faculty of Computer Science and Control Engineering, Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Beijing 100045, China (e-mail: yi.pan@siat.ac.cn).

Digital Object Identifier 10.1109/TMC.2023.3284994

complexity, and the tendency to obtain only local optimal solutions.

3) Last, in the existing researches on the placement of MEC servers, the interference management factor is less considered. However, addressing the issue of interference becomes an urgent problem that needs to be solved, particularly when multiple users simultaneously request MEC resources.

This article proposes an algorithm based on Graph Partition and Multiple Choice-UCB to address these challenges. For the interference management problem, finding the maximal independent set of undirected graphs is a feasible solution. Specifically, in a network graph consisting of nodes representing servers, a maximal independent set defines a set of servers which can operate in parallel without interference [6]. In this article, we abstract the MEC servers in a graph as vertices, and the collisions between MEC servers as connecting edges. By identifying the maximal independent set in this graph, we can determine a subset of MEC servers that do not interfere with each other in the given scenario. We posit that MEC servers corresponding to isolated vertices in the graph do not cause interference with other MEC servers.

The Multi-Armed Bandits (MAB) paradigm is a promising solution for MEC server placement. In each round, the objective is to choose a feasible action to take certain arm and gets the corresponding reward, from the chosen arm solely without any additional reward information for un-chosen actions (other arms). The goal is to maximize its cumulative (total) reward over all rounds. To achieve this goal, algorithm must balance exploration and exploitation; that is, algorithm must make choice between the action with the current highest average reward and the action with the potential highest average reward [7]. One of the most widely used strategies for stochastic multi-armed bandits is the Upper Confidence Bound algorithm, which is based on the optimistic principle in the face of uncertainty [8], [9]. In this article, we take the placement scheme of the MEC servers as the arms. If we define required number of MEC servers from current system as M, then the decision to placement plan of M MEC servers equals to shaking M arms.

The main contributions of this article are summarized as follows:

- *Graph Partition-based interference management:* We model the interference management problem into an undirected graph problem with each MEC server as vertices and mutual interference between paired MEC servers as edges. An edge exists if the euclidean distance between the MECs is less than the sum of the coverage radii of each other. For each component of the overall undirected graph, we obtain subsets of MEC servers that do not interfere each other.
- *Multiple Choice-UCB (MC-UCB) algorithm:* A multiple choice-UCB algorithm is proposed to solve the problem that multiple arms need to be selected in one round. In this article, we consider the servers placement in the paradigm of stochastic MAB. Instead of shaking one arm in each round, we shake M arms in each selection, *i.e.*, deciding the placement scheme of M edge computing servers. To speed up the selection, we prune one arm with the lowest current UCB value after a certain number of rounds. If the arm is more likely to be smaller than the arm with the Mth highest UCB value, we prune the arm from the candidate set early.
- *QoS evaluation indicators and simulation results:* Transmission delay, throughput, and user density are comprehensively considered in the evaluation of system service quality. For the problem of MEC server placement, we comprehensively consider the location, bandwidth, transmission rate, coverage radius of the MEC servers in the scenario. Then we use the average transmission delay, total throughput and average reward as indicators to compare and evaluate the proposed algorithms with the baseline algorithms. The simulation results demonstrate the superiority of the proposed algorithms.

The rest of the article is organized as follows: Section II summarizes the related work. Section III presents the system model and further defining performance indicators of multi-user and multi-MEC server. Section IV presents the flow of Graph Partition algorithm and MC-UCB algorithms. Section V simulates the proposed algorithm and baseline algorithms under different performance indicators. Section VI summarizes our work and draw the conclusion.

II. Related Work

In this section, we study previous related work on interference management and MEC server placement, while we analyze the limitations of existing work.

Related Work on Interference Management: For the problem of interference management in MEC environment, Xiao et al. used secure reinforcement learning to avoid choosing a risk offloading strategy that could not meet the task computing delay requirement [10]. Sha et al. modeled the entire network as a graph. Based on graph theory, they proposed a minimum beam collision algorithm and proved that the algorithm could obtain the global minimum beam collision solution [11]. For multi-cell multi-user channel allocation, Quan et al. obtained an optimization problem with controllable complexity by grouping weakly interfering units and assigning resources among these possible interference patterns in the network [12]. Yan et al. proposed an adaptive interference management method based on three redesigned techniques, which can be applied to sparse rural and dense urban networks, according to the location of the terminal, interference type, QoS requirements, and processing capability [13].

In addition, the interference management between MEC servers can be abstract as solving the problem of maximal independent sets in an undirected graph. The algorithms for finding maximal independent sets can be classified into two main categories: distributed algorithms and centralized algorithms. Regarding distributed algorithms, Moscibroda et al. studied the distributed complexity of computing maximum independent sets (MIS) in wireless networks with completely unknown topology, asynchronous wake-up, and no collision detection mechanism [14]. In another study [15], the authors addressed the resource allocation problem for device-to-device (D2D)

communication in cellular systems. However, in the aforementioned distributed algorithms, the authors only identified a single maximal independent set.

As for centralized algorithms, Johnson et al. proposed an algorithm that generates all the maximal independent sets of graphs in lexicographic order, with only a polynomial delay between the outputs of two consecutive independent sets [16]. Tsukiyama et al. introduced an effective algorithm for generating all the maximal independent sets using a depth-first search in a "dynamic" binary tree [17]. Leung et al. provided efficient algorithms for generating all maximal independent sets in interval graphs, circular-arc graphs, and chordal graphs. However, these algorithms are not applicable for solving maximal independent set problems in general graphs [18].

Related Work on Placement of MEC Servers: The placement of edge computing servers had also attracted widespread attention from researchers. Li et al. proposed an adaptive clustering algorithm based on AP suitability evaluation to solve the problem of edge server placement, which minimized the task cost of computing tasks [19]. In another article [20], the authors formulated the MEC server placement problem as a multi-objective optimization problem and designed a particle swarm optimization-based energy-aware edge server layout algorithm. Kasi et al. formulated the low latency and workload balancing requirements in edge server placement strategies as a multi-objective constrained optimization problem, using genetic programming and local optimization algorithms (hill climbing and simulated annealing) to solve the optimal solution [21]. Zhang et al. proposed a comprehensive process that combines edge server and service placement. They presented a two-step method involving a clustering algorithm and nonlinear programming to tackle this problem [22]. Cui et al. formally modeled the robust edge server location problem and proposed an integer programming-based optimization method to find the optimal solution [23]. Also for the robust edge server location problem, Qu et al. mathematically formulated the RSP problem as a robust max-min optimization form, and proposed a polynomial-time algorithm with a better approximation ratio [24]. In addition, the combinatorial multi-armed bandit algorithm offers a promising approach for addressing MEC server deployment decisions. Chen et al. established a comprehensive framework for a specific category of large-scale Combinatorial Multi-Armed Bandit (CMAB) problems, which involves grouping simple arms with unknown distributions into super arms [25]. Meanwhile, Gai et al. simultaneously monitor all selected random variables and generate a reward by forming a linear weighted combination of these variables [26]. However, these methods cannot be directly applied to interference-avoidance-based MEC server deployment problems.

Limitations of Previous Researches: While previous studies have demonstrated the superiority of their proposed algorithms, there are still limitations in the existing work: 1) Insufficient consideration of interference management factors in the deployment of MEC: Although there have been numerous research efforts discussing the problem of interference management and MEC server placement, the consideration of interference management factors in the deployment of MEC is still inadequate. 2) Neglect of hardware configuration and other factors in MEC server deployment: Previous research on MEC server deployment has predominantly focused on the geographical distribution of MEC servers, while overlooking the influence of MEC server hardware configuration (such as bandwidth and transmit power) and other pertinent factors on the MEC server deployment problem. In this article, our objective is to improve the quality of service in the MEC system by addressing both interference management and edge server placement problems. We take into account the hardware configuration, geographic distribution, and surrounding user density of each MEC server. Through simulation results, we validate the feasibility of our proposed algorithm in scenarios where these factors vary.

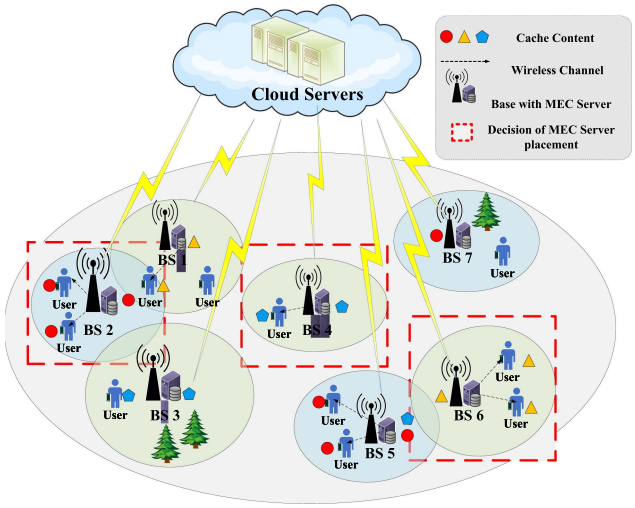


Fig. 1. System model.

III. System Model

In this section, we first analyze the system model of multi-user multi-MEC server, and then we comprehensively consider transmission delay, throughput, and user density to define the quality of service of the system.

A. Multi-User Multi-MEC Server Model

As shown in Fig. 1, we define a multi-user multi-MEC server communication system. The set of users in this scenario is defined as $\mathcal{I} = \{1, 2, \ldots, I\}$. The set of MEC server candidates is defined as $\mathcal{N} = \{1, 2, \ldots, N\}$. For each MEC server $n \in \mathcal{N}$ in the candidate set, we denote the MEC server n by a quadruple $(P^n_{ser}, L^n_{ser}, B^n_{ser}, D^n_{ser})$. Where, P^n_{ser} represents the transmit power of MEC server n, L^n_{ser} represents the location coordinates of MEC server n, B^n_{ser} represents the bandwidth of MEC server n, and D^n_{ser} represents the maximum coverage radius of MEC server n. In this scenario, the hardware configuration and geographic coordinates of each MEC server in the candidate set are different. We assume that users in the scenario have similar service requirements, such as watching sports videos. At the beginning of each time slot, the cloud server sends multiple video resources to the MEC server, which then

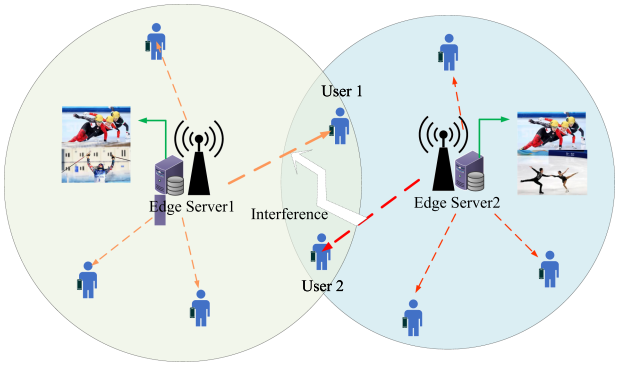


Fig. 2. Model of interference from transmission link collision.

TABLE I
SYSTEM PARAMETERS

Parameters	Explanation
$\mathcal{I} = \{1, 2, \cdots, I\}$	set of users
$\mathcal{N} = \{1, 2, \cdots, N\}$	set of MEC servers
$Data_k$	data size of cache resource k
K	number of components of undirected graph G
$\mathcal{A} = \{1, 2, \cdots, N\}$	set of all arms
a_i	the arm with the i-th largest UCB

caches the received content. The cached content is subsequently broadcasted to the surrounding users. As shown in Fig. 1, colored entities are cached content. When multiple users request cache resources from the same MEC server simultaneously, the MEC server broadcasts its cached content to all users requesting its services. As a result, there is no interference among users requesting cache resources from the same MEC server. However, when two MEC servers with overlapping coverage areas $(\text{distance}(MEC_i, MEC_j) < (D^i_{ser} + D^j_{ser}))$ transmit video resource simultaneously. Users in overlapping areas may experience transmission link collisions, thus causing interference.

As shown in Fig. 2, user 1 requests cache resources of MEC server 1, and meanwhile user 2 requests cache resources of MEC server 2. Due to their proximity, the communication link between User 1 and User 2 becomes interfered, resulting in a degradation of communication quality for both users. To maximize the service quality for users in the system, at the beginning of each time slot, we aim to select several MEC servers that do not interfere with each other to cache video resources.

The system parameters are shown in Table I.

B. Assessment of Quality of Service

In this section, we evaluate the quality of service of the system from three aspects of video resource transmission delay, throughput, and user density. At the end of this section, the definition formula of the quality of service of the system is given.

1) Transmission Delay: Let $T^{k,tran}_{n,i,t}$ denote the transmission delay of the cache resource k from MEC n to user i at time t. $T^{k,tran}_{n,i,t} = \frac{Data_k}{R_{n,i}}$, where $Data_k$ is the data size of the video resource, $R_{n,i}$ is the sending rate of MEC server n sending tasks to user i, which is defined as:

$$R_{n,i} = B^n_{ser} \log_2 \left(1 + \frac{h^2 P^n_{ser} d^{-\theta}_{i,n}}{N_0}\right) \tag{1}$$

In the above formula, B^n_{ser} is the bandwidth of the MEC server n, h is the fading factor of the signal, P^n_{ser} is the transmission power of the MEC server n, and $d_{i,n}$ is the transmission distance of the video resource transmitted by the MEC server n to the user i. θ is a constant, N_0 is Gaussian white noise, and N_0 satisfies the normal distribution. Let $d_{i,n} = \sqrt{(L^n_x - L^i_x)^2 + (L^n_y - L^i_y)^2}$, indicating the transmission distance from MEC server n to user i. Wherein, L^n_x and L^n_y represent the abscissa and ordinate of MEC server n, and L^i_x and L^i_y represent the abscissa and ordinate of user i. At the same time, we define a threshold τ for the upper limit of the transmission delay of video resources. When the user's transmission delay is greater than the threshold τ, we consider the video transmission to fail.

Assuming that at time t, N_{total} video task requests are generated in the coverage of MEC server n, we define the average transmission delay of all video resources within the coverage of MEC server n as $T_n = \frac{\sum_{k=1}^{N_{\text{total}}} T^{k,tran}_{n,i,t}}{N_{\text{total}}}$.

2) Throughput: We define throughput as the number of video tasks successfully transmitted by MEC server n per unit time, namely: $Thr_n = \frac{N_{\text{succ}}}{t}$. Where, N_{succ} $(N_{\text{succ}} \leq N_{\text{total}})$ is the number of successfully transmitted video tasks, and t is the size of the unit time slot. The level of the throughput is closely related to the calculation rate and service range of MEC server, and at the same time, throughput reflects the workload and resource utilization of the MEC server n.

3) User Density: Define user density as $Dens_n = \frac{N_{\text{people}}}{\pi(D^n_{\text{ser}})^2}$ where N_{people} is the number of users within the coverage of MEC server n, and $\pi(D^n_{\text{ser}})^2$ is the MEC server coverage of n. When the density of users around the MEC server n is large, it means that the MEC server n can provide services for more users in the system, so the existence of the MEC server is of great significance.

4) Quality of Service: Suppose that at time t, there are N_{people} users within the coverage of MEC server n who have generated N_{total} tasks. Among them, N_{succ} tasks are successfully transferred. The calculation formula of the service quality of users within the coverage of MEC server n is: $QoS_n = \alpha T_n + \beta Thr_n + \gamma Dens_n$. Where, α, β, γ are weight factors, which represent the proportion of transmission delay, throughput, and user density in the service quality evaluation of MEC server n, respectively. Among them, T_n is negatively correlated with QoS_n, the greater the transmission delay, the worse the service quality of the MEC server n is. Also we need to notice that the QoS_n fluctuates in a small range due to the uncertainty of the channel state. In other words, QoS_n is randomly generated within a reward distribution when testing the service quality of MEC server n. In addition, we map QoS_n to between $0 \sim 1$ for better data normalization.

In this article, we define the reward of the system as $R = \sum_{n=1}^{N} QoS_n \mathbb{I}_n$, where $\mathbb{I}_n$ is the indicator vector, when the MEC server is selected, $\mathbb{I}_n$ returns 1, otherwise it returns 0. Taking the

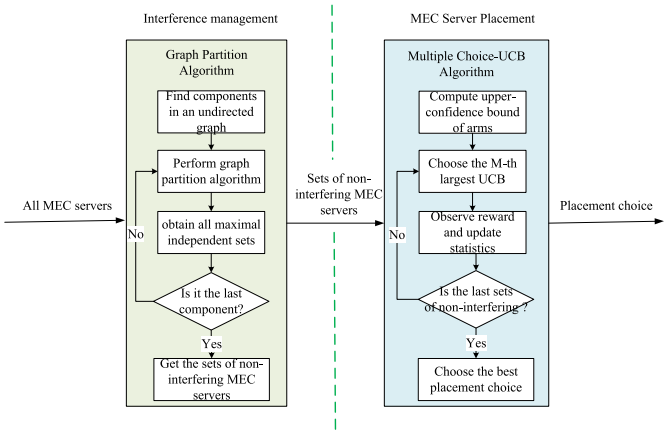


Fig. 3. Flow of GP-MCUCB algorithm.

service quality (reward) of the system as the goal, we formulate the objective function as:

$$\begin{aligned} &\max E\left[\sum_{n=1}^{N} QoS_n \mathbb{I}_n\right] \\ &\text{s.t. } C_1 : \mathbb{I}_n \in \{0,1\} \\ &\quad C_2 : \sum_{n=1}^{N} \mathbb{I}_n = M \\ &\quad C_3 : d_{i,j} < \left(D_{\text{ser}}^{i} + D_{\text{ser}}^{j}\right) \\ &\qquad d_{i,j} = \text{ distance } (MEC_i, MEC_j)\, \mathbb{I}_i \mathbb{I}_j \end{aligned} \tag{2}$$

The constraint condition C_1 indicates the value of the indicator vector, C_2 indicates that only M MEC servers can be selected from the candidate set $\mathcal{N}$ to deploy in the current scene, and C_3 indicates that there is no interference between the selected MEC servers.

The objective function is a nonlinear integer programming problem. The main difficulties in solving this problem are: 1) The channel quality in the MEC scenario changes dynamically, and the return value of QoS_n has volatility. Dynamic system scenarios place higher demands on decision-making algorithms. 2) As the number of MECs deployed in the scenario increases, the decision space increases linearly, and the computational complexity of constraint C3 increases exponentially.

IV. Graph Partition and MC-UCB Algorithm

In this article, our objective is to maximize the quality of service (QoS) of the system by selecting an optimal subset of non-interfering MEC servers to cache video resources at the beginning of each time slot. However, the selection process becomes challenging due to the varying hardware configurations and locations of the MEC servers. To address this problem, we propose the Graph Partition and MC-UCB (GP-MCUCB) algorithm. The overall flow of the algorithm is depicted in Fig. 3. First, we employ the Graph Partition algorithm to identify all sets of non-interfering MEC servers in the current scenario.

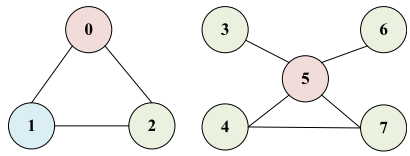


Fig. 4. Undirected graph G.

This step ensures that the MEC servers deployed in the scenario do not interfere with each other, providing a foundation for subsequent decision-making. Next, these non-interfering subsets of MEC servers are passed as input variables to the Multiple Choice-UCB (MC-UCB) algorithm. The MC-UCB algorithm is responsible for selecting the optimal deployment decision within each non-interfering subset and generating the corresponding output. Finally, the algorithm selects the deployment decision with the highest reward value as the final deployment solution.

Next, we will introduce the Graph Partition and Multiple Choice-UCB algorithms respectively.

A. Graph Partition-Based Interference Management Algorithm

First, we solve the problem of interference avoidance between MEC servers, which means we need to find all non-interfering subsets among the N MEC servers in the scenario.

We traverse the MEC servers in the scenario and build an $N \times N$ adjacency matrix A. If there is an intersection between the coverage areas of MEC server i and MEC server j, there is interference, then the element of the matrix A at $< i, j >$ is 1, otherwise it is 0. We define that there is no interference between the servers themselves, so the elements on the diagonal are 0. Next, we use the MEC servers as vertices, and the conflict relationships between the MEC servers as edges. Then we obtain the interference model graph G according to the adjacency matrix A. As shown in Fig. 4, MEC servers that do not interfere with other MEC servers are isolated vertices. MEC servers with the interference relationship form the connected component. This graph representation helps us identify the groups of MEC servers that cannot be placed together due to interference.

Then, we need to find all components in the undirected graph G corresponding to the adjacency matrix A. Suppose that the undirected graph G contains K components. We perform the graph partition algorithm on them in turn.

The core idea of graph partition algorithm is to construct a spanning tree based on the adjacency list of the complement graph, and traverse all paths from the root node to the leaf nodes, outputting all independent sets. From these independent sets, we select all maximal independent sets as the final results. The overall process of the algorithm is a backtracking algorithm based on pruning. Specifically, to minimize the search time, we made improvements to the backtracking algorithm based on two key intuitions: First, we can quickly prune nodes that are more likely to cause interference with other nodes, reducing unnecessary search efforts. Second, based on the content of the complement graph's adjacency list, we narrow down the range of successor nodes for each node. Specifically, each node's successor nodes

TABLE II
EDGE SETS AND COMPLEMENTARY EDGE SET OF COMPONENT 1

Edge sets
$[0, 1]$
$[0, 2]$
$[1, 2]$

TABLE III
EDGE SETS AND COMPLEMENTARY EDGE SET OF COMPONENT 2

Edge sets	Complementary edge set
$[3, 5]$	$[3, 4]$
$[4, 5]$	$[3, 6]$
$[4, 7]$	$[3, 7]$
$[5, 6]$	$[4, 6]$
$[5, 7]$	$[6, 7]$

include the empty node and nodes that do not conflict with it. This approach significantly speeds up the search process and improves the success rate of finding maximal independent sets. The adjacency list of complement graph and the details of the algortithm are are defined as follows:

To obtain the adjacency list of the complement graph, we first find all the edge (u, v) in the undirected graph G, which is called Edge set. The set of edges that exist in the complement graph but not in Edge set is called the complementary edge set (CES). Based on CES, we can easily generate the adjacency list of the complement graph in lexicographic order.

Next, we will construct a spanning tree based on the adjacency list of the complement graph. We define the successor nodes of the initial node as all the nodes in the graph. The successor nodes of non-root nodes are empty nodes and other nodes that are not interfering with them. Empty nodes have no successor nodes. This step can be easily determined by querying the adjacency list of the complement graph.

Once the spanning tree is constructed, we search for all paths from the root node to the leaf nodes. If all nodes in a path do not have interference, we consider it a valid path and store all the nodes in that path into an independent set. For a maximal independent set, adding any vertex that is not already included would make it no longer an independent set. Therefore, we need to remove any subsets that exist in the obtained independent sets to obtain the maximal independent sets as output.

In order to explain the Graph Partition algorithm more clearly, we introduce an example to describe the process of the algorithm. Suppose that the undirected graph G is shown as Fig. 4. We can see that there are 2 components.

We process each component in Fig. 4 in turn. For component 1 that contains three nodes numbered 0, 1, and 2. There is an interference relationship between any two nodes, as shown in Table II. Therefore, the CES table of component 1 is an empty set, all nodes have only empty node as their successor node, and they form a maximal independent set that contains only themselves.

For component 2 that contains five nodes numbered 3, 4, 5, 6, and 7, the corresponding CES table is shown in Table III. The successor nodes of the initial node S are [5, 4, 7, 3, 6]. The successor nodes of node 3 are [4, 7, 6, ∅]. The successor nodes of node 4 are [6, ∅]. The successor node of node 5 is [∅]. The

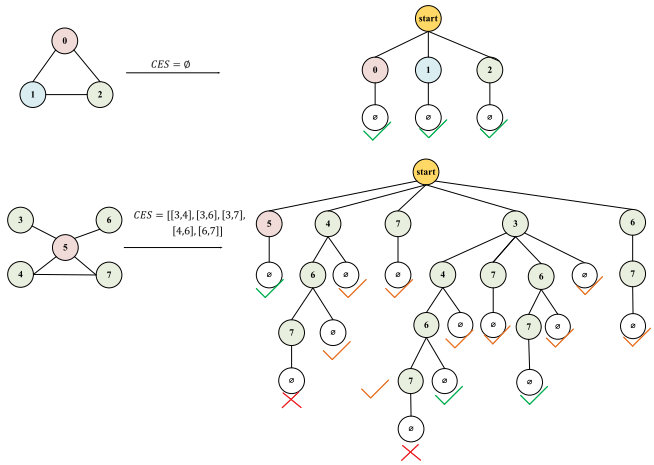


Fig. 5. Graph partition algorithm.

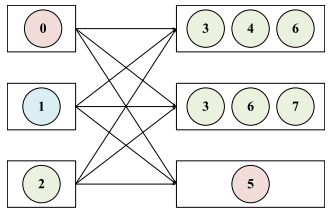


Fig. 6. Sets of non-interfering MEC servers.

successor nodes of node 6 are [7, ∅]. The successor node of node 7 is [∅]. The path from the root node to the leaf node will form an independent set, so we get independent sets [5], [4, 6], [4], [7], [3, 4, 6], [3, 4], [3, 6, 7], [3, 6], [3, 7], [3], [6, 7], [6]. Among them, the maximal independent sets are [5], [3, 4, 6], [3, 6, 7]. The process of constructing the spanning tree is shown in Fig. 5.

Combine the maximal independent set obtained by component 1 and component 2. Finally, as shown in Fig. 6, the maximal independent set corresponding to the undirected graph G is obtained [0,3,4,6], [0,3,6,7], [0,3,7], [1,3,4,6], [1,3,6,7], [1,3,7], [2,3,4,6], [2,3,6,7], [2,3,7].

B. MC-UCB Based MEC Server Placement Algorithm

After obtaining the subsets of non-interfering MEC servers, for each subset, we need to select M MEC servers that provide the highest reward in terms of quality of service. If the number of MEC servers that do not interfere with each other in the subset is less than M, we will skip the subset. Finally, we select a set of MEC servers with the highest reward from the results of all subsets. We analyze this problem from the perspective of MAB. Assuming that there is a global agent in the system as a gambler, and N MEC servers in the subset that do not interfere with each other are N arms, the gambler shakes M arms in each round, and the gambler's overall goal is to minimize accumulated regret in a limited number of rounds. "Regret" is defined as the difference between the reward of continuously selecting the optimal arm and the reward obtained in the actual

algorithm [27], [28]. Formally, the regret of the previous round can be obtained as:

$$R(T) = u^* \cdot T - \sum_{t=1}^{T} u\left(a_t\right) \tag{3}$$

Wherein, $u(a) = E[D_a] = \mathrm{E}[\overline{u_t}(a)]$ represents the expected reward of the arm under the reward distribution D_a, $u^* = \max_{a \in A} u(a)$, represents the best average reward among the average rewards generated by all arms.

We let $\mathcal{A}$ denote the set of all arms, $\mathcal{A} = \{1, 2, \ldots, N\}$, $a \in \mathcal{A}$. $N_t(a)$ represents the number of times arm a is selected by the algorithm in rounds $1 \sim t$, $\overline{u_t}(a)$ represents the average reward of arm a estimated by the previous t rounds. The reward of arm a is defined as: $R(a) = QoS_a = \alpha T_a + \beta Thr_a + \gamma Dens_a$, $QoS_a \in [0, 1]$, according to Hoeffding's inequality [29]:

$$\begin{aligned} \Pr\left[\left|\overline{u_t}(a) - u(a)\right| \leq r_t(a)\right] &\geq 1 - \tfrac{2}{T^4} \\ r_t(a) &= \sqrt{\tfrac{2\log(T)}{N_t(a)}} \end{aligned} \tag{4}$$

where T is a fixed parameter representing the entire time domain in which the algorithm runs. Because $1 - \frac{2}{T^4}$ is a large number, it can be considered that $\left|\overline{u_t}(a) - u(a)\right| \leq r_t(a)$ is a high probability event, namely $\overline{u_t}(a)$ has a greater probability of being within the confidence interval $[u(a) - r_t(a), u(a) + r_t(a)]$, and similarly $u(a)$ has a greater probability of being within the confidence interval $[\overline{u_t}(a) - r_t(a), \overline{u_t}(a) + r_t(a)]$.

The algorithm flow of MC-UCB is is shown in Algorithm 1.

In the initialization phase, the algorithm shakes each arm once, and gets the random reward R_a fed back by the arm a. Then the algorithm enters T round loop.

In each round, the MC-UCB algorithm selects the arm with the highest M value of $UCB_t(a)$. Gambler execute these arms in turn, get corresponding rewards and update these arms. When the current round t is greater than the threshold $T_{\max}$, before the end of round t, we judge whether the arm N with the lowest current $UCB_t(N)$ value is smaller than the lower limit of the confidence interval of the arm with the M-th highest value of $UCB_t(M)$. If it is, there is a high possibility that the arm N is not among the arms with the highest M in the $UCB_t(N)$ value, and the arm will be deleted from the candidate set.

Next, we analyze the regret upper bound of the MC-UCB algorithm: Suppose that the arms with the top M highest UCB_t form a set $\mathcal{M}$. If the arm $a_i \notin \mathcal{M}$, $a_M \in \mathcal{M}$ and the current time t gets $UCB_t(a_i) > UCB_t(a_M)$, then regret occurs. According to the above analysis of the high probability situation, we can infer: $u(a_i) + r_t(a_i) \geq \overline{u_t}(a_i), \overline{u_t}(a_M) + r_t(a_M) \geq u(a_M)$. At the same time, since $UCB_t(a_i) > UCB_t(a_M)$, it can be deduced that $\overline{u_t}(a_i) + r_t(a_i) > \overline{u_t}(a_M) + r_t(a_M)$. Arranging the above formula can get:

$$\begin{aligned} & u\left(a_i\right) + 2r_t\left(a_i\right) \\ \geq\ & \overline{u_t}\left(a_i\right) + r_t\left(a_i\right) \\ >\ & \overline{u_t}\left(a_M\right) + r_t\left(a_M\right) \\ \geq\ & u\left(a_M\right) \end{aligned} \tag{5}$$

Algorithm 1: MC-UCB Algorithm.

Input: Number of rounds T
Output: Accumulated rewards
for $a = 1, 2, \ldots N$ **do**
  $N_0(a) = 1$
  $\overline{u_0}(a) = R_a$
end for
for $t = 1, 2, \ldots T$ **do**
  for $a = 1, 2, \ldots N$ **do**
    Compute upper-confidence bound of arm a
    $UCB_t(a) = \overline{u_t}(a) + \sqrt{\frac{2\log(t)}{N_t(a)}}$
  end for
  Sort N arms in descending order by $UCB_t(a)$ value
  for $i = 1, 2, \ldots M$ **do**
  Choose the action a_i=the arm with the i-th largest UCB_t
  Observe reward $R_{i,t} \sim v(a_i)$
  Update statistics for action a_i:
  $N_t(a_i) = N_{t-1}(a_i) + 1$,
  $\overline{u_t}(a_i) = \frac{\overline{u_{t-1}}(a_i) \times N_{t-1}(a_i) + R_{i,t}}{N_t(a_i)}$
end for
if $t > T_{\max}$ and $N > M$ **then**
  if $UCB_t(N) < \overline{u_t}(M) - \sqrt{\frac{2\log(t)}{N_t(M)}}$ **then**
    Abandon arm N, $N = N - 1$
    if $N == M$ **then**
      break
    end if
  end if
end if
end for

From this we can deduce:

$$\begin{aligned} \Delta\left(a_i\right) &= u\left(a_M\right) - u\left(a_i\right) \\ &\leq 2r_t\left(a_i\right) \\ &= 2\sqrt{\frac{2\log(T)}{N_t(a)}} \end{aligned} \tag{6}$$

The cumulative regret obtained by taking action a_i in the first t rounds is the product of $\Delta(a_i)$ and the number of times a_i is selected in the first t rounds, namely:

$$\begin{aligned} R\left(t; a_i\right) &= N_t\left(a_i\right) \times \Delta\left(a_i\right) \\ &\leq N_t\left(a_i\right) \cdot O\left(\sqrt{\frac{2\log(T)}{N_t\left(a_i\right)}}\right) \\ &= O\left(\sqrt{N_t\left(a_i\right) \cdot \log(T)}\right) \end{aligned} \tag{7}$$

The cumulative regret $R(t)$ for the first t rounds is the cumulative sum of regrets arising from taking all non-optimal actions,

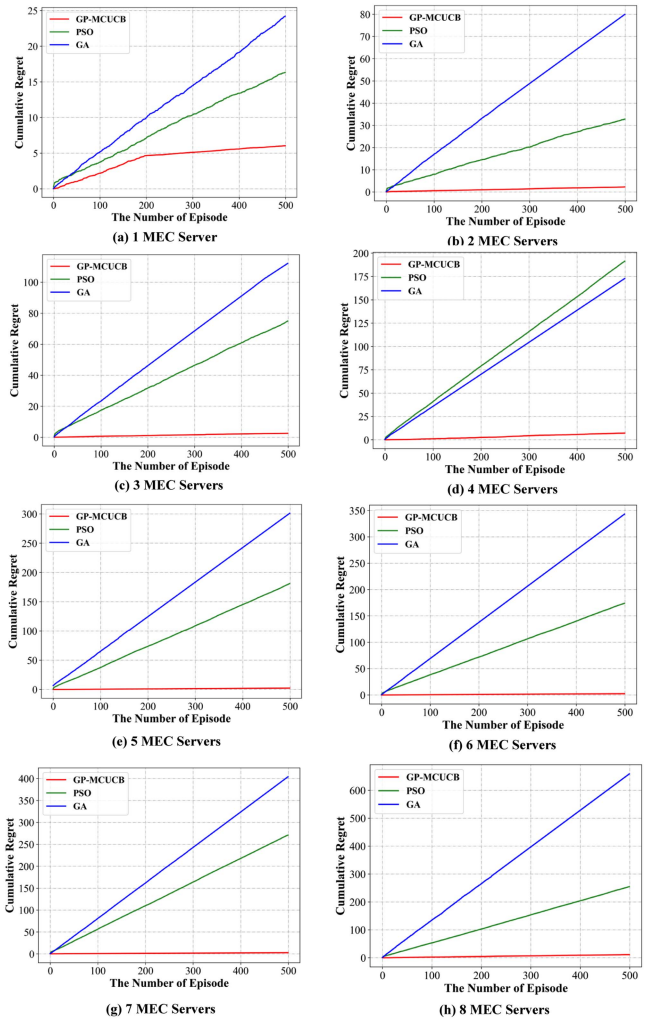


Fig. 7. Cumulative regret comparison under different MEC servers.

namely:

$$R(t) = \sum_{a \in \mathcal{A} \wedge a \notin \mathcal{M}} R(t;a) \leq \sum_{a \in \mathcal{A} \wedge a \notin \mathcal{M}} \sqrt{N_t(a)} \cdot O(\sqrt{\log(T)}) \tag{8}$$

By analyzing the above formula, it is known that the maximum number of regrets in each round is $\min\{M, K-M\}$, and the maximum number of regrets in the previous t rounds is $t \times \min\{M, K-M\}$. When we consider $N_t(a)$ as a variable, $\sqrt{N_t(a)}$ is a convex function, which can be obtained according to JenSen's inequality:

$$\frac{1}{k} \sum_{a \in \mathcal{K} \wedge a \notin \mathcal{M}} \sqrt{N_t(a)} \leq \sqrt{\frac{1}{k} \sum_{a \in \mathcal{K} \wedge a \notin \mathcal{M}} N_t(a)} \leq \sqrt{\frac{t \times \min\{M, K-M\}}{k}} \tag{9}$$

Bring it into Formula 8 to get: $R(t) \leq O(\sqrt{\log(T)}) \times \sqrt{t \times k \times \min\{M, K-M\}} \leq O(\sqrt{t \log(T)})$. Let "hp" means the event of high probability, "lp" means the event of low probability. The expectation of $R(t)$ is:

$$\begin{aligned} E[R(t)] &= E[R(t) \mid hp] \times pr \times [hp] \\ &\quad + E[R(t) \mid lp] \times pr \times [lp] \\ &\leq O(\sqrt{t \log(T)}) \times 1 + 1 \times O\left(T^{-4}\right) \\ &= O(\sqrt{t \log(T)}) \end{aligned} \tag{10}$$

In summary, the logarithmicity of the upper bound of regret for the MC-UCB algorithm has been proved.

TABLE IV
SIMULATION PARAMETERS

Parameters	Value
Channel Bandwidth B_{ser}^{n}	$8 \sim 10$ MHz
Transmit Power P_{n}^{ser}	$100 \sim 300$ W
Coverage Radius R_{n}^{ser}	$250 \sim 300$ m
Pathloss Parameters θ	4
Tolerate delay of the task τ	5 us
Weight factor α	0.4
Weight factor β	0.2
Weight factor γ	0.4

V. SIMULATION RESULTS

In this section, we compare the proposed GP-MCUCB algorithm with the baseline algorithms. We deployed 50 MEC servers in the scenario, and each MEC server varies in bandwidth, coverage radius, and geographic location. 1000 terminal users are discretely distributed around the 50 MEC servers, and the user density around each MEC server varies. We set the baseline algorithms as Particle Swarm Optimization (PSO) [20] and Genetic Algorithm (GA) [21]. In PSO algorithm, all particles have a fitness value determined by a reward function and determine the direction and distance of exploration through their speed. The particles determine the global optimum by searching in parallel. Genetic algorithm is a method of searching for the optimal solution by simulating the natural evolution process. With the help of the genetic operators of natural genetics, crossover and mutation to generate a population representing a new solution set. In order to adapt to the scenario of this article, we set the search dimension of the PSO and GA algorithms to be 50, and return a negative reward to suppress the occurrence of the situation when the selected result does not satisfy the constraints of (2). The simulation parameters are shown in Table IV.

A. Evaluation of Cumulative Regret

We first analyze the GP-MCUCB algorithm and the baseline algorithms with cumulative regret as the evaluation indicator. As mentioned in (3), cumulative regret is the difference between the reward value of continuously selecting the optimal decision and the reward value actually obtained by the current algorithm. The numerical size and upward trend of cumulative regret reflect the performance of the algorithm. In Section IV-B, we established the logarithmic upper bound for the regret of the proposed GP-MCUCB algorithm. Now, we aim to validate this bound through simulations. By conducting simulations, we can

TABLE V
IDs OF THE DEPLOYED MEC SERVERS UNDER DIFFERENT MEC SERVERS

algorithms / MEC server(s)	GP-MCUCB	PSO	GA
1	[14]	[22]	[32]
2	[14,31]	[31, 33]	[27, 33]
3	[14, 31, 32]	[22, 27, 29]	[24, 31, 39]
4	[14, 31, 32, 33]	[16, 18, 38, 44]	[2, 16, 29, 40]
5	[14, 31, 32, 33, 29]	[2, 24, 31, 32, 48]	[21, 22, 44, 46, 47]
6	[14, 29, 31, 32, 33, 49]	[2, 14, 16, 31, 44, 48]	[3, 15, 32, 34, 39, 48]
7	[14, 31, 32, 33, 29, 49, 44]	[2, 16, 24, 26, 31, 32, 48]	[3, 11, 12, 16, 26, 34, 43]
8	[14, 31, 32, 33, 29, 49, 44, 3]	[2, 7, 11, 14, 18, 24, 29, 49]	[3, 19, 26, 37, 39, 40, 46, 49]

TABLE VI
DATA OF EXPERIMENT UNDER DIFFERENT USERS

algorithms / the number of MEC servers	1	2	3	4	5	6	7	8
Average transition delay of GP-MCUCB algorithm (us)	**4.8972**	**3.6426**	**3.5932**	**3.3201**	**3.0462**	**2.7873**	**2.8185**	**2.6702**
Total Throughput of GP-MCUCB algorithm	609	1420	2260	3119	4062	5122	5946	6929
Average reward of GP-MCUCB algorithm	10.8898	13.3073	14.2735	15.076	15.713	16.528	16.5282	16.9451
Average transition delay of PSO algorithm (us)	6.5792	5.4601	5.139	4.2751	4.351	4.235	3.7196	3.2472
Total Throughput of PSO algorithm	589	1275	1936	2714	3289	4018	4912	5797
Average reward of PSO algorithm	10.2171	11.4477	11.3531	13.2229	12.5443	12.9304	13.568	14.0558
Average transition delay of GA algorithm (us)	6.225	5.7724	5.841	4.588	4.048	4.1039	3.1241	2.8564
Total Throughput of GA algorithm	568	1084	1709	2473	3759	4600	5731	6265
Average reward of GA algorithm	9.6738	9.3108	10.0819	11.4044	14.2403	14.8568	15.7896	15.201

observe the behavior of the algorithms in practice and verify their performance.

It can be seen from Fig. 7 that the cumulative regret of the GP-MCUCB algorithm rises more slowly than the baseline algorithms. In most cases, the cumulative regret of the GA algorithm rises the fastest, followed by the PSO algorithm. We can see that the cumulative regret of the GA algorithm rises almost linearly. This is because the GA algorithm tends to get stuck in a local optimum, making the same decisions in subsequent iterations. Thus, each cumulative regret increases at a near constant rate.

Moreover, it is evident that the cumulative regret values of the GA and PSO algorithms increase noticeably as the number of deployed MEC servers in the scenario grows. This phenomenon can be attributed to the exponential growth in the search dimension of the GA and PSO algorithms, which occurs with the increasing number of MEC servers. Consequently, this exacerbates the limitations of traditional heuristic algorithms when dealing with larger-scale MEC server deployment scenarios. In contrast, the GP-MCUCB algorithm exhibits superior performance. Based on the simulation results mentioned above, it is apparent that the upper bound of cumulative regret remains below 15.

B. Performance Comparison With Different MEC Servers

In this section, we compare the performance of the GP-MCUCB algorithm with the PSO and GA algorithms using the average transmission delay, total throughput, and average reward as indicators. The numerical results of the simulation are shown in Tables V and VI.

We conducted an analysis comparing the GP-MCUCB algorithm with baseline algorithms using the average transmission delay as the metric. The results, depicted in Fig. 8, indicate

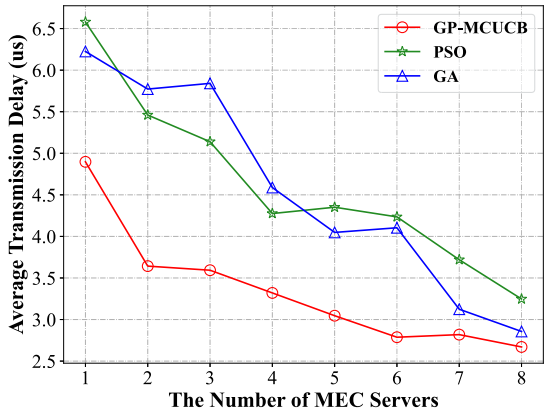


Fig. 8. Average transmission delay under different MEC servers.

that all three algorithms exhibit a decreasing trend as the number of deployed MEC servers in the scenario increases. The reason behind this trend is that with a greater number of MEC servers, users have more opportunities to request cache resources from servers that are in closer proximity to them. This shorter transmission distance between users and MEC servers leads to reduced transmission delays for video resources. From the Fig. 8, we can see that as the number of MEC servers increases, the average processing delay decreases from fast to slow. Compared with the baseline algorithm, the GP-MCUCB algorithm has obvious advantages.

As depicted in Fig. 9, both the GP-MCUCB algorithm and the baseline algorithm show a significant increase in throughput as the number of deployed MEC servers in the scenario increases. This phenomenon can be attributed to the fact that a larger number of MEC servers in the scenario provides users with more choices for selecting servers. Consequently, users have a higher probability of requesting services from MEC servers that are

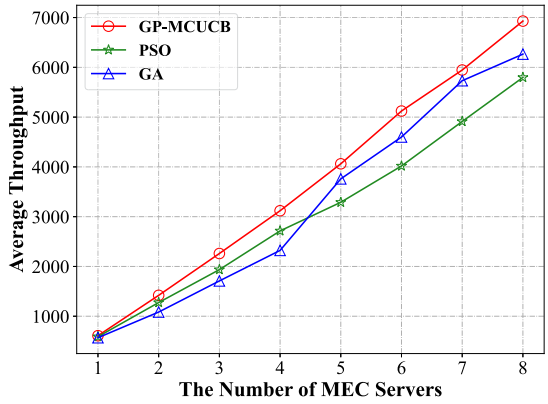


Fig. 9. Total throughput delay under different MEC servers.

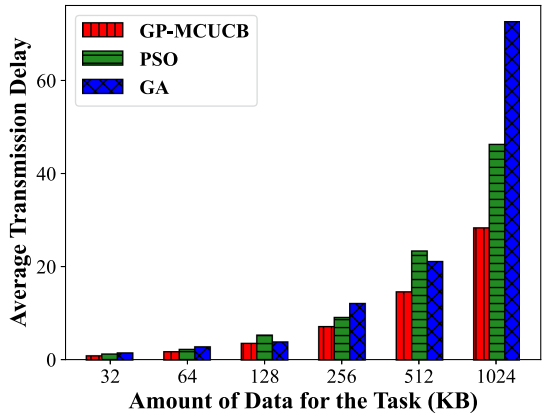


Fig. 11. Average transmission delay under different data size.

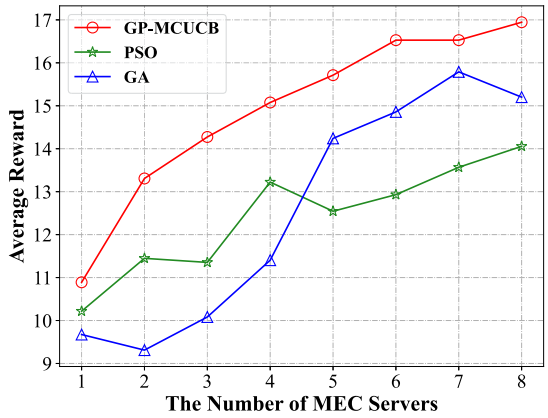


Fig. 10. Average reward under different MEC servers.

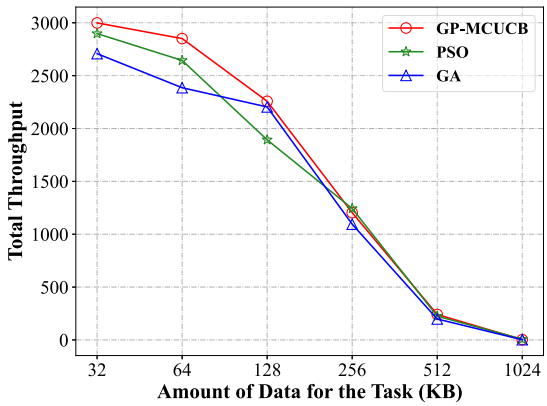


Fig. 12. Total throughput under different data size.

in closer proximity to them, resulting in reduced transmission time for video resources. This, in turn, enables more video tasks to be completed within the minimum time threshold, thereby improving system throughput.

The simulation results clearly demonstrate that the GP-MCUCB algorithm outperforms both the PSO algorithm and the GA algorithm.

As depicted in Fig. 10, there is a noticeable increase in the average reward of tasks as the number of deployed MEC servers in the scenario increases. This can be attributed to the comprehensive consideration of various factors in our design of the average reward metric, including the average transmission delay of tasks, throughput of video tasks, and the user density around MEC servers. User density serves as a crucial criterion for evaluating the effectiveness of MEC server placement, but its value is solely determined by the coordinates of the current user and the coverage range of MEC servers. It does not undergo significant changes with variations in the number of deployed MEC servers in the scenario.

Based on the preceding analysis, we can conclude that as the number of MEC servers increases, the average transmission delay of tasks decreases, and the overall throughput improves, leading to an increase in the average reward of tasks. Fig. 10 illustrates that the GP-MCUCB algorithm outperforms the PSO algorithm by more than 6%, while the GA algorithm demonstrates an improvement of over 4% in performance.

C. Performance Comparison With Different Data Size

Next, we analyze the difference between the performance of the GP-MCUCB algorithm and the baseline algorithms under different data sizes. The numerical results of the simulation are shown in Table VII.

As shown in Fig. 11, as the amount of video resource data in the scenario increases, the average transmission delay of video resources increases. From the figure, it is evident that the GP-MCUCB algorithm consistently outperforms both the PSO algorithm and the GA algorithm. Notably, the advantage of the GP-MCUCB algorithm becomes more pronounced as the data volume of the video resources increases. These findings highlight the superior performance of the GP-MCUCB algorithm in effectively managing and allocating resources, particularly in scenarios with larger data volumes.

As depicted in Fig. 12, both the GP-MCUCB algorithm and the baseline algorithm exhibit a downward trend in total throughput as the amount of video resource data in the scenario increases. This is primarily attributed to the increase in average transmission delay, which leads to a higher number of video

TABLE VII
DATA OF EXPERIMENT UNDER DIFFERENT DATA SIZE

algorithms \ data size of the task (KB)	32	64	128	256	512	1024
Average transition delay of GP-MCUCB algorithm (us)	0.895	1.772	3.561	7.1683	14.6003	28.3185
Total Throughput of GP-MCUCB algorithm	2999	2851	2258	1204	241	3
Average reward of GP-MCUCB algorithm	**20.2795**	**18.942**	**14.2731**	**5.8035**	**-3.5893**	**-10.6632**
Average transition delay of PSO algorithm (us)	1.2728	2.278	5.3307	9.1206	23.3531	46.2039
Total Throughput of PSO algorithm	2898	2644	1893	1243	225	7
Average reward of PSO algorithm	20.019	17.5666	11.1981	5.4896	-7.2463	-17.8399
Average transition delay of GA algorithm (us)	1.5161	2.8302	3.8728	12.1095	21.1028	72.4949
Total Throughput of GA algorithm	2708	2386	2204	1095	198	2
Average reward of GA algorithm	18.0096	15.8351	13.8016	3.3562	-6.1594	-28.1213

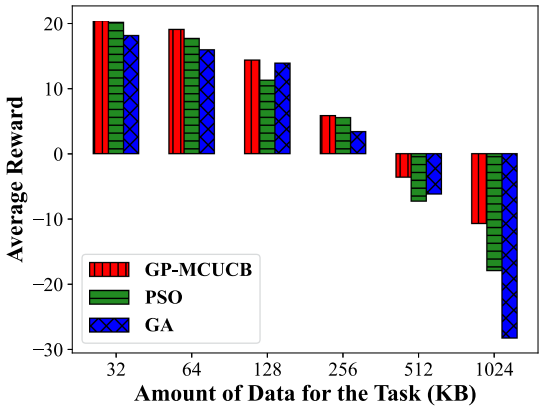


Fig. 13. Average reward under different data size.

tasks failing to meet the delay threshold, consequently reducing the overall throughput. Moreover, the figure highlights that as the amount of video resource data increases, the decline rate of total throughput in the scenario becomes more prominent. Overall, when compared to the baseline algorithms, the GP-MCUCB algorithm demonstrates superior performance.

As shown in Fig. 13, as the amount of video resource data in the scenario increases, under all algorithms, the average reward shows a downward trend. This is because the user density around the MEC server does not change much when the number of MEC servers deployed in the scenario remains the same. However, the average transmission delay tends to increase and the total throughput tends to decrease as the volume of video resource data grows. Consequently, the average reward experiences a downward shift. In comparison to the GA and PSO algorithms, the GP-MCUCB algorithm demonstrates clear advantages.

VI. CONCLUSION

In this article, we propose an algorithm that combines Graph Partition and MC-UCB to address the challenge of managing edge server interference and optimizing their placement in a multi-user multi-MEC server scenario, aiming to enhance the overall service quality of the system. The interference between MEC servers is represented using an undirected graph in the graph partition algorithm, which allows us to obtain several candidate sets of MEC servers with no interference among them. Subsequently, the MC-UCB algorithm is employed to make optimal decisions regarding the placement of a fixed number of MEC servers.

To evaluate the performance of our proposed algorithm, we compare it with existing PSO and GA algorithms in terms of average reward, average transmission delay, and total throughput. The simulation results clearly demonstrate that our GP-MCUCB algorithm outperforms the other approaches in all evaluated metrics. As part of our future work, we intend to further optimize the algorithm's performance by exploring the impact of the number of MEC servers and other time-varying factors. This will enable us to enhance the overall efficiency and effectiveness of the algorithm in real-world scenarios.

REFERENCES

[1] M. Muniswamaiah, T. Agerwala, and C. C. Tappert, "A survey on cloudlets, mobile edge, and fog computing," in *Proc. IEEE 8th Int. Conf. Cyber Secur. Cloud Comput. 7th IEEE Int. Conf. Edge Comput. Scalable Cloud*, 2021, pp. 139–142.

[2] C. Wang, F. R. Yu, C. Liang, Q. Chen, and L. Tang, "Joint computation offloading and interference management in wireless cellular networks with mobile edge computing," *IEEE Trans. Veh. Technol*, vol. 66, no. 8, pp. 7432–7445, Aug. 2017.

[3] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, Feb. 2018.

[4] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[5] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, "Toward edge intelligence: Multiaccess edge computing for 5G and Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020.

[6] F. Kuhn, T. Moscibroda, T. Nieberg, and R. Wattenhofer, "Fast deterministic distributed maximal independent set computation on growth-bounded graphs," in *Proc. Int. Symp. Distrib. Comput.*, Springer, 2005, pp. 273–287.

[7] R. Watanabe, J. Komiyama, A. Nakamura, and M. Kudo, "KL-UCB-based policy for budgeted multi-armed bandits with stochastic action costs," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 100-A, no. 11, pp. 2470–2486, 2017.

[8] H. W. J. Reeve, J. Mellor, and G. Brown, "The k-nearest neighbour UCB algorithm for multi-armed bandits with covariates," *Proc. Mach. Learn. Res.*, in Firdaus Janoos, Mehryar Mohri, and Karthik Sridharan, editors, PMLR, 2018, pp. 725–752.

[9] S. Hashima, M. M. Fouda, Z. M. Fadlullah, E. M. Mohamed, and K. Hatano, "Improved UCB-based energy-efficient channel selection in hybrid-band wireless communication," in *Proc. IEEE Glob. Commun. Conf.*, 2021, pp. 1–6.

[10] L. Xiao, X. Lu, T. Xu, X. Wan, W. Ji, and Y. Zhang, "Reinforcement learning-based mobile offloading for edge computing against jamming and interference," *IEEE Trans. Commun.*, vol. 68, no. 10, pp. 6114–6126, Oct. 2020.

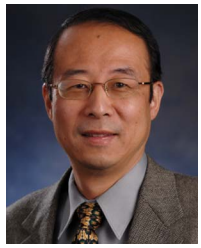
[11] Z. Sha, Z. Wang, S. Chen, and L. Hanzo, "Graph theory based beam scheduling for inter-cell interference avoidance in mmWave cellular networks," *IEEE Trans. Veh. Technol*, vol. 69, no. 4, pp. 3929–3942, Apr. 2020.

[12] Q. Kuang and W. Utschick, "Energy management in heterogeneous networks with cell activation, user association, and interference coordination," *IEEE Trans. Wireless Commun.*, vol. 15, no. 6, pp. 3868–3879, Jun. 2016.
[13] S. Yan, X. Cao, Z. Liu, and X. Liu, "Interference management in 6G space and terrestrial integrated networks: Challenges and approaches," *Intell. Converged Netw.*, vol. 1, no. 3, pp. 271–280, 2020.
[14] T. Moscibroda and R. Wattenhofer, "Maximal independent sets in radio networks," in *Proc. 24th Annu. ACM Symp. Princ. Distrib. Comput.*, 2005, pp. 148–157.
[15] A. Köse and B. Özbek, "Resource allocation for underlaying device-to-device communications using maximal independent sets and knapsack algorithm," in *Proc. IEEE 29th Annu. Int. Symp. Pers. Indoor Mobile Radio Commun.*, 2018, pp. 1–5.
[16] D. S. Johnson, M. Yannakakis, and C. H. Papadimitriou, "On generating all maximal independent sets," *Inf. Process. Lett.*, vol. 27, no. 3, pp. 119–123, 1988.
[17] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM J. Comput.*, vol. 6, no. 3, pp. 505–517, 1977.
[18] J. Y.-T. Leung, "Fast algorithms for generating all maximal independent sets of interval, circular-arc and chordal graphs," *J. Algorithms*, vol. 5, no. 1, pp. 22–35, 1984.
[19] B. Li, P. Hou, H. Wu, R. Qian, and H. Ding, "Placement of edge server based on task overhead in mobile edge computing environment," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 9, 2021, Art. no. e4196.
[20] Y. Li and S. Wang, "An energy-aware edge server placement algorithm in mobile edge computing," in *Proc. IEEE Int. Conf. Edge Comput.*, 2018, pp. 66–73.
[21] S. K. Kasi et al., "Heuristic edge server placement in industrial Internet of Things and cellular networks," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10308–10317, Jul. 2021.
[22] X. Zhang, Z. Li, C. Lai, and J. Zhang, "Joint edge server placement and service placement in mobile edge computing," *IEEE Internet Things J.*, vol. 9, no. 13, pp. 11261–11274, Jul. 2022.
[23] G. Cui, Q. He, X. Xia, F. Chen, H. Jin, and Y. Yang, "Robustness-oriented k edge server placement," in *Proc. IEEE/ACM 20th Int. Symp. Cluster Cloud Internet Comput.*, 2020, pp. 81–90.
[24] Y. Qu et al., "Server placement for edge computing: A robust submodular maximization approach," *IEEE Trans. Mobile Comput.*, vol. 22, no. 6, pp. 3634–3649, Jun. 2023.
[25] W. Chen, Y. Wang, and Y. Yuan, "Combinatorial multi-armed bandit: General framework and applications," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2013, pp. 151–159.
[26] Y. Gai, B. Krishnamachari, and R. Jain, "Combinatorial network optimization with unknown variables: Multi-armed bandits with linear rewards and individual observations," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, pp. 1466–1478, Oct. 2012.
[27] C.-W. Lee, H. Luo, C.-Y. Wei, and M. Zhang, "Bias no more: High-probability data-dependent regret bounds for adversarial bandits and MDPs," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 15522–15533.
[28] S. Ito, S. Hirahara, T. Soma, and Y. Yoshida, "Tight first- and second-order regret bounds for adversarial linear bandits," in *Proc. Adv. Neural Inf. Process. Syst. Annu. Conf. Neural Inf. Process. Syst.*, 2020, pp. 2028–2038.
[29] V. Moulos, "A hoeffding inequality for finite state Markov chains and its applications to Markovian bandits," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 2777–2782.

Zheyu Zhao (Graduate Student Member, IEEE) is currently working toward the master's degree in computer science and technology with the University of Science and Technology of China. Her research interests include cloud/edge computing, algorithm design, and intelligent IoT.

Hao Cheng is currently currently working toward the PhD degree in EEMCS of Technische Universiteit Delft. His research interest includes machine learning algorithm and corresponding application, especially in non stationary low SNR area.

Xiaohua Xu received the bachelor's degree from the Zhu Kezhen College of Zhejiang University and the PhD degree from Illinois Institute of Technology, USA. He is a professor with the School of Computer Science and Technology, University of Science and Technology of China. His research interests are edge computing, algorithm design, and intelligent IoT.



Yi Pan (Senior Member, IEEE) is current the dean and chair professor of Faculty of Computer Science and Control Engineering with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China. He has been a regents' professor emeritus and served as the chair of Computer Science Department with Georgia State University during 2005–2020. He has also served as an interim associate dean and chair of Biology Department during 2013–2017. He joined Georgia State University in 2000, was promoted to full professor in 2004, named a Distinguished University professor in 2013, and designated a regents' professor (the highest recognition given to a faculty member by the University System of Georgia) in 2015. His work has been cited more than 16 000 times based on Google Scholar and his current H-index is 82. He has published more than 450 papers, including more than 100 papers in IEEE/ACM transactions/journals and edited/authored 43 books. His current research interests mainly include parallel and cloud computing, wireless networks, and bioinformatics.