Blockchain-enabled Multiple Sensitive Task-offloading Mechanism for MEC Applications

Yang Xu, *Member, IEEE,* Hangfan Li, Cheng Zhang, Zhiqing Tang, *Member, IEEE,* Xiaoxiong Zhong,*Member, IEEE,* Ju Ren, *Senior Member, IEEE,* Hongbo Jiang, *Senior Member, IEEE,* and Yaoxue Zhang, *Senior Member, IEEE*

Abstract—As mobile devices proliferate and mobile applications diversify, Mobile Edge Computing (MEC) has become widely adopted to efficiently allocate computing resources at the network edge and alleviate network congestion. In the MEC initial phase, the absence of vital information presents challenges in devising task-offloading policies, and identifying malicious devices responsible for providing inaccurate feedback is complex. To fill in such gaps, we introduce a consortium blockchain-enabled Committee Voting based Task Offloading Model (CVTOM) to collaboratively formulate resource allocation policies and establish deterrence against malicious servers producing erroneous results intentionally. Different voting principle mechanisms of each committee member are first designed in a Blockchain-enabled system which helps to represent the system's resource status. Additionally, we propose a Multi-armed Bandits related Thompson Sampling based Adaptive Preference Optimization (TSAPO) algorithm for task-offloading policy, enhancing the timely identification of potent edge servers to improve computing resource utilization which first considers dynamic edge server space and parallel computing scenarios. The solid proof process greatly contributes to the theoretical analysis of the TSAPO. The simulation experiments demonstrate the delay and budget can be reduced by around 25% and 10% respectively, showcasing the superior performance of our approach.

Index Terms—Mobile Edge Computing, Blockchain, Task-offloading, Multi-armed Bandits.

1 Introduction

Mobile devices and applications, such as virtual reality software [1], are changing how people engage with technology, enabling decentralized, efficient, and reliable P2P interactions, and opening doors for innovation and entrepreneurship. The significant increase in delay-sensitive or budget-sensitive tasks due to the expanding use of mobile devices and various built-in applications has prompted the adoption of Mobile Edge Computing (MEC) in numerous areas [2]. A stable MEC system contains information on all network servers, such as task processing and information transmission capabilities, authentication information, etc. This enables the central server to create secure and effective task-offloading policies [3]. However, in the initial stage of the MEC system, this information is unknown, making it crucial to develop an efficient task allocation policy in such a scenario.

Security issues arise from the centralized structure of MEC. In many existing MEC networks, centralized task-offloading centers handle service requests and task-offloading policies. A significant risk is the random selection of an edge server for task-offloading without proper authentication and security certification, leading to severe consequences if the server is hacked or compromised. Moreover, some bad actors exploit the system by providing misleading task processing results that are difficult to track.

Efficiency issues, at the same time, stem from the lack of critical information about connected servers, such as edge servers' task processing capacity and channel conditions. Without these key parameters [4, 5], task-offloading policies struggle to accurately assign tasks to suitable servers, potentially leading to delay-sensitive tasks being allocated to servers with limited capacity.

Cryptographic techniques provide effective solutions for addressing ***security issues***. While these techniques enhance privacy protection in a centralized framework, the encryption and decryption processes involve significant information transmission, consuming plenty of communication resources. Therefore, a decentralized communication framework, like integrating a committee mechanism through a consortium blockchain, is better suited to address this issue. Blockchain-enabled MEC networks make responsibility tracing feasible, acting as a strong deterrent against malicious attackers [6, 7]. Through a voting mechanism, the offloading policies implemented by the smart contract in the system are reliable, guaranteeing transparency in task execution throughout the network [8–11].

The Multi-armed Bandits (MAB) framework is an effective approach to ***efficiency issues*** [12]. It is a type of rein-

- *Y. Xu, H. Li, C. Zhang, and H. Jiang are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China. E-mail: xuyangcs@hnu.edu.cn; HFLee@hnu.edu.cn; zhangchengcs@hnu.edu.cn; hongbojiang2004@gmail.com (Corresponding author: Hangfan Li.)*
- *Z. Tang is with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, Zhuhai 519087, China. E-mail: zhiqingtang@bnu.edu.cn*
- *X. Zhong is with the Cyberspace Security Research Center, Peng Cheng Laboratory, Shenzhen 518000, China. E-mail: xixzhong@gmail.com*
- *J. Ren and Y. Zhang are with the Department of Computer Science and Technology, BNRist, Tsinghua University, Beijing 100084, China. E-mail: renju@tsinghua.edu.cn; zhangyx@tsinghua.edu.cn*

forcement learning algorithm that involves balancing exploration and exploitation in a sequential decision-making process to identify the most optimal options (a.k.a. arms, actions, or servers).

However, three new challenges have emerged when implementing these solutions. The **first challenge** arises in the context of MEC, where online/offline servers introduce changes to the action space within the MAB framework [13–15]. Traditional MAB algorithms do not account for these variations [16]. Factors such as low battery levels or the movement of mobile devices in and out of service range can lead to fluctuations in the number of servers. In such scenarios, a more adaptable algorithm capable of adjusting to real-time changes may be necessary. The **second challenge** occurs when dealing with a large volume of computing tasks, where traditional MAB algorithms sequentially offload tasks. These algorithms adjust task-offloading policies only after receiving feedback from the previous action, resulting in considerable time consumption. For instance, in the Upper Confidence Bound (UCB) algorithm [17–19], formulating the next task-offloading policy before obtaining feedback from the previous one may lead to the repetition of the same choice, which is highly undesirable. The **third challenge** arises when addressing diverse sensitive tasks. Existing bandit-based works primarily concentrate on optimizing a single target, such as delay or budget.

Motivations. To address the above ***security issues*** and three new challenges from ***efficiency issues***, we first develop a consortium blockchain-enabled Committee Voting-based task-offloading Model (CVTOM) to devise task-offloading policies while preventing malicious servers jointly. **Different voting principles mechanism is first designed, helping committee members to vote for a task-offloading policy according to the status of its resources which could reflect the resource situation of the system, and tasks can be assigned accordingly.** Secondly, we propose an MAB-related Thompson Sampling-based Adaptive Preference Optimization (TSAPO) algorithm to guide computing tasks offloading. **Different from other MAB-based algorithms, dynamic server space is first considered, helping a committee member to win the vote which recreates the scenario of a blockchain node online/offline realistically.** Finally, we conduct simulation experiments to demonstrate the functionality of the CVTOM system. The superior performance of the TSAPO algorithm is validated through theoretical analysis and practical experiments. The contributions of this paper are summarized as follows:

- We build a CVTOM MEC system, integrating a consortium blockchain-based committee mechanism and a voting consensus approach that can mitigate the central server's risks of paralysis or malicious intent. When confronted with different task-offloading policies, a voting process is utilized to select the policy with the expected optimal performance. ***We are the first to set different voting principles for committee members based on their resource strengths and weaknesses, considering both delay and budget simultaneously.***
- We proposed the TSAPO algorithm for the CVTOM system to optimize task-offloading policies, ***considering dynamic server space over traditional MAB-based algorithms.*** Its advantages include 1) Time and budget savings, aiding the master edge server in achieving consensus via parallel task allocation voting; 2) Rapid identification of the top-performing edge server. Leveraging the Thompson Sampling algorithm, it dynamically adjusts task allocation policies per round based on feedback; 3) Minimization of resource wastage during exploration by applying upper confidence bound principles to filter out servers with notably poor performance.
- ***A highly informative proof process is given.*** We conduct theoretical analysis to show that the proposed TSAPO has strict upper bounds on regret, with linear growth over time. ***Its upper bound is*** $\sqrt{\lambda T\left(M+N\right)\log\left(\lambda T\right)}$ ***where*** T ***is the number of system running rounds,*** $M+N$ ***is the maximum number of the connected servers,*** λT ***is the number of tasks collected in CVTOM system***. We perform simulation experiments to demonstrate the applicability and superiority of TSAPO compared to traditional bandit algorithms in terms of optimal server-choosing rate, total delay, total budget cost, and task-dropping rate.

The paper is structured as follows: Section 2 recaps existing research in related fields. Section 2.3 introduces a comprehensive CVTOM model for MEC and outlines optimization challenges. In Section 4, the TSAPO algorithm is introduced with a detailed mathematical analysis of its regret convergence. Section 5 evaluates the performance of TSAPO through simulated experiments. Finally, Section 6 concludes by summarizing the key findings and contributions of the paper.

2 Related Work

In this section, notable progress has been made in addressing both security and *efficiency issues* in MEC task-offloading. For security, integrating blockchain into the MEC system offers a robust solution. For efficiency, approaches with and without critical network state information, such as task parallelism offloading, optimizing multiple objectives, and employing dynamic action spaces in MAB-based models, have proven effective during task offloading stages. Details on each approach are summarized in Table 1.

2.1 Addressing MEC Task-offloading Security Issues

For *security issues*, some previous studies have demonstrated strong performance in defending against and detecting malicious task-offloading servers in MEC.

Dong *et al.* [27] introduce a Fusion algorithm to detect servers using low-quality models or deviating from protocols by comparing public sample results. Ghodsi *et al.* [28] propose the SafetyNets to verify cloud server inference performance, while Zhao *et al.* [29] present Veri-machine-learning for fair payments in machine learning services. All rely on a trustworthy center, which can be compromised in certain scenarios.

Blockchain-enabled networks could offer a reliable communication environment, with smart contracts facilitating

TABLE 1
Summary of the Existing Works

Architectural Attributes	Existing Works					Our Work
	[7, 8, 20, 21]	[22, 23]	[24, 25]	[13–15, 20, 21]	[21, 26]	
Blockchain-enabled MEC System	✓	✓				✓
Task Parallelism Offloading		✓	✓			✓
Multiple Optimization Objectives			✓		✓	✓
Dynamic Action Space				✓		✓

interactions among IoT devices [7, 8, 20]. Xu *et al.* [20] propose BeCome for blockchain-enabled MEC system which is a computation offloading method for edge computing, addressing resource imbalance and security vulnerabilities while optimizing resource allocation through genetic algorithms and decision-making policies. Following his work, Guo *et al.* [7] propose a blockchain-enabled Vehicloak system that successfully verifies transactions without revealing location data in a vehicle scenario. And Nguyen *et al.* [8] using block mining through a Proof-of-Reputation consensus mechanism to propose a similar method that considers more network aspects when optimizing system utility via distributed deep reinforcement learning, game-theoretic approaches and voting mechanism.

2.2 Addressing MEC Task-offloading Efficiency Issues with Critical Network Information

For *efficiency issues*, an efficient task-offloading policy is crucial for optimizing network resource utilization and enhancing service quality [30]. Task parallelism offloading and multiple optimization objectives are among the most widely considered methods.

Several studies focus on minimizing delay, energy, or others in distributed learning networks [4, 21, 24–26]. Wang *et al.* [4] enhances the robustness of task-offloading algorithms in systems considering server mobility and remaining battery power. Li *et al.* [21] investigates how to jointly optimize sensor node deployment, edge node deployment, data routing, and data offloading to minimize the number of deployed sensor nodes, the number of deployed edge nodes, and event reporting delay and maximize network lifetime. Zhu *et al.* [26] optimize edge computing service provider profits and energy in a blockchain-enabled MEC system.

Task parallelism offloading is also a useful way to increase the efficiency of the network. Shi *et al.* [22] propose a deep reinforcement learning vehicle-to-vehicle task-offloading algorithm to offload part of a task to a neighbor edge server, allowing them to process tasks parallelly in a VEC scenario. In Guo *et al.* [23] work, different collaborative mining networks could jointly interact with network resource demanders where task processing efficiency and mining network profits are increased.

Combining both multiple optimized targets and task parallelism offloading is also a well-researched plant. He *et al.* [24] consider minimizing performance and operational expenditures while maintaining system performance at a predetermined level by parallelism offloading tasks. Li *et al.* [25] propose HASP which enhances multi-NN performance through task parallelism offloading and multiple optimized targets by isolating resources and fine-tuning allocation.

However, as the system scale grows, it becomes challenging to obtain or predict system information, particularly channel conditions [31]. Addressing task-offloading in a system with unpredictable key factors or lacking prior knowledge is a significant challenge.

2.3 Addressing MEC Task-offloading Efficiency Issues without Critical Network Information

MAB framework [32] offers a promising approach to meet such requirements. Ye *et al.* [33] have introduced a decentralized task-offloading algorithm that operates within an epoch-based framework to optimize participants' rewards concerning system delay. These fundamental principles are also applied in vehicular edge computing networks [34] and fog networks [35]. This policy facilitates the offloading of user tasks to different servers in an online learning environment.

In some scenarios, like systems with battery-enabled edge servers or mobile servers, the connected edge server set is dynamic Therefore, dynamic action spaces for the task-offloading algorithm makers are also essential [13–15, 20, 21].

Gao *et al.* [13] propose the eAUCB algorithm to handle the situation that some arms may be unavailable in some rounds and the arms will bid inconsistently in different rounds. Hong *et al.* [14] propose off-policy learning to offload tasks with logged bandit feedback to the connected servers. Zhu *et al.* [15] analyze algorithms that address the contextual bandits problem, demonstrating strong empirical performance when the number of possible actions is small. However, providing guarantees for decision-making in large, continuous action spaces remains challenging, highlighting a significant gap between theory and practice. Despite the shortcomings of their work, it still inspires us to design task-offloading algorithms.

3 System Model

In this section, we introduce CVTOM within a blockchain environment with dynamically varying task-sensitive conditions in MEC. The model will be discussed in the subsequent sequence: (1) System Components; (2) System Operation Process; (3) Model Details; (4) Problem Formulation.

3.1 System Components

The following are explanations about the main components, actions, and essential definitions in the CVTOM which definitions are shown in Table 2.

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2024.3507153

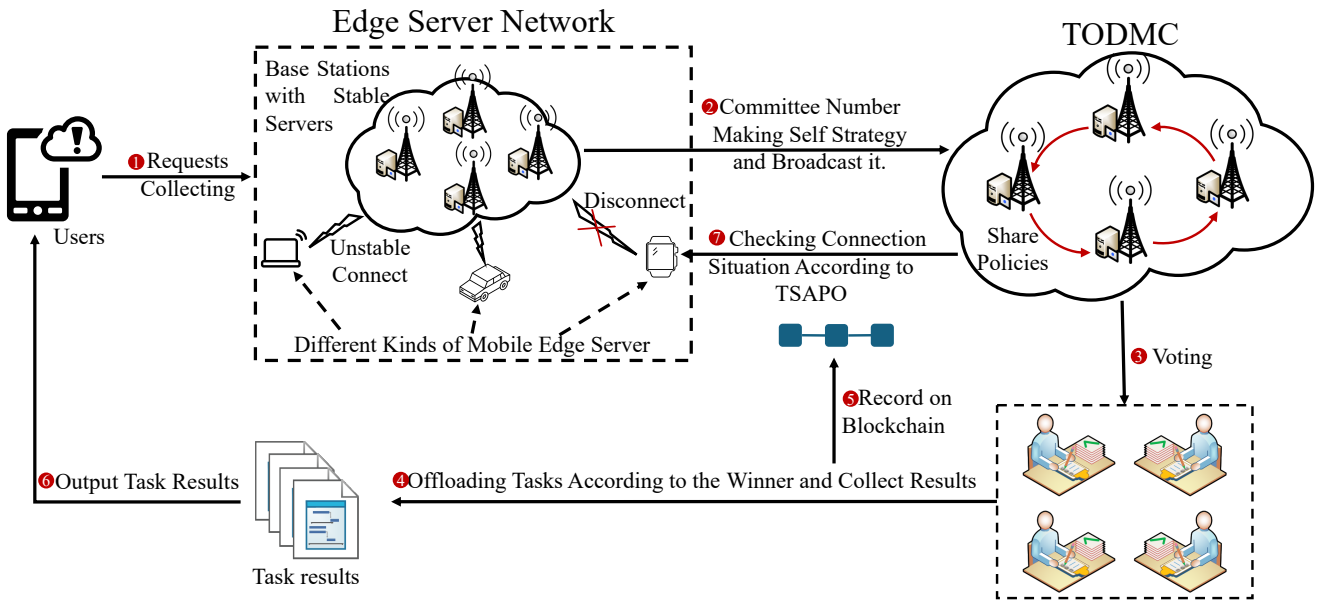


Fig. 1. System Model.

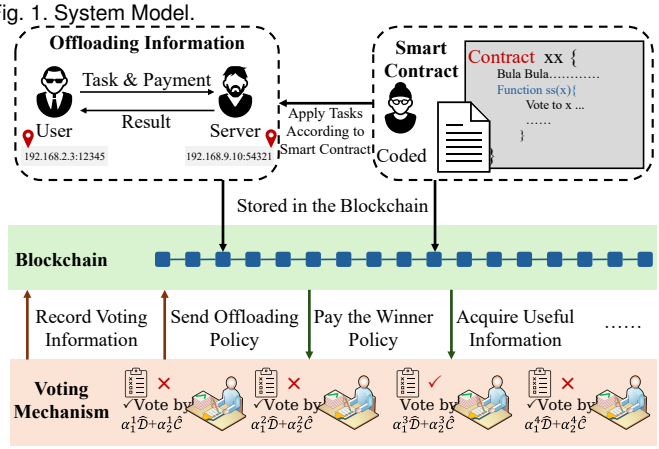


Fig. 2. Voting and Recording Information on Blockchain. In Voting Mechanism, $\alpha_1^1, \alpha_2^1 \sim \alpha_1^4, \alpha_2^4$ are preference parameters of 1∼4 committee members. $\hat{\mathcal{D}}, \hat{\mathcal{C}}$ in this figure is the delay and budget performance discrepancy in Eq. (7) which is different from the detailed definition in Section 3.3.2.

Mobile Devices (Users). They are portable devices such as laptops, smartphones, etc. Software applications, like VR games, on mobile devices often produce time-sensitive computing tasks while in use. Occasionally, these tasks may be managed by the devices, but for the purposes of this paper, we are excluding this possibility. We assume that the devices discussed here lack the capability to promptly handle the tasks they create. Therefore, all tasks will be offloaded to edge servers for processing.

Tasks & Requests. Mobile devices initially create service requests and then transmit tasks to the designated edge server based on the task-offloading policy. The request size is minimum and insignificant. The quantity of tasks received in each round follows a Poisson distribution.

Edge Servers. In our system, the edge servers $\mathcal{S}$ are categorized into two types: stable servers and temporary servers. Stable servers, represented by **base stations** (Fig. 1), are equipped with powerful computers. These base stations handle task transportation and communication for task-offloading policies, while the powerful computers process delay-sensitive tasks. Stable servers remain online continuously.

Temporary servers are either migration devices or small servers powered by a battery, capable of temporarily processing certain tasks. These temporary servers may be offline temporarily or permanently. To simplify, we refer to the first M servers in $\mathcal{S}$ as stable servers $\mathcal{S}_{sta}$ and the remaining servers as temporary servers $\mathcal{S}_{tem}$, with the server set $\mathcal{S}$ defined as $\mathcal{S} = \mathcal{S}_{sta} \cup \mathcal{S}_{tem}$.

Rounds. Consider this system as progressing through a round horizon $\mathcal{T}$. A single round t signifies the smallest cycle in the system. Within a round, service requests are first collected, followed by the implementation of task allocation policies, and finally, tasks are executed while evaluating policy performance.

TODMC. The system operates on a blockchain network with edge servers as blockchain nodes. Node behaviors, including task processing details, are stored on the blockchain to ensure the credibility of the system. Within our system, certain nodes possess robust task processing capabilities and rapid information exchange speeds (capable of swift consensus). These nodes, known as **powerful nodes**, can be either stable servers or temporary servers with high processing power and close proximity to other nodes. Task-offloading algorithms in the form of smart contracts are deployed on these powerful nodes. Grouped together, some powerful nodes form a unique task-offloading Decision Making Committee (TODMC), which is responsible for devising task-offloading policies and voting for an accounting node.

Policy. A policy π is defined as a set of ("Task", "Target server") pairs that are round-dependent.

Voting Function. The functions $u(\pi)$ are created to evaluate the effectiveness of a policy by assessing its performance. Voting functions are configured as convex combinations of delay and cost, consistently differing from other members in the TODMC.

Preference: It determines the voting function for each TODMC member, denoted as a tuple of argu-

ments. Preferences may include time-saving, cost-saving, energy-saving, etc.

3.2 System Operation Process

To ensure that the servers are trustworthy, we establish CVTOM within an MEC system. The details will be explained in the following three parts: 1) CVTOM advantages over traditional blockchain-enabled MEC; 2) Initial steps of CVTOM; 3) Operation process of CVTOM.

CVTOM distinguishes itself from other systems by assuming that the computing tasks it generates may have dual sensitivity, incorporating both delay and cost sensitivity. It's noteworthy that the weighting of the voting function for delay versus budget can vary among individuals, a significant departure from prior research [22]. These intricate details will be extensively modeled in the forthcoming subsection.

In a distributed communication MEC scenario, depicted in Fig. 1, the initial steps of our system are as follows:

1) Select a committee of nodes with exceptional task processing capabilities. Members are chosen based on their exceptional past performance or mutual recommendations among nodes, with the assumption that all committee members were predetermined from the start.
2) Apply task-offloading algorithms on smart contracts and create voting functions for committee members according to their specific preferences, such as energy efficiency or time optimization.

Then, the operation process for a single round of CVTOM is as follows:

1) Edge servers collect computing requests roundly.
2) For each committee node, based on the algorithm in the smart contract (which is unique for each individual), proposes its own task-offloading policy and broadcasts it.
3) Receive policies from other members, predict their outcomes, and vote for the best-performing policy based on their own voting function. Detailed in Fig. 2, the winner is the third committee member.
4) The node with the highest number of votes becomes the bookkeeping node and applies its task-offloading policy.
5) As Fig. 2 shown, record the policy performance result to the blockchain including the task, result, and payments of each user.
6) Output the task results to users.
7) Checking server connection status by the TSAPO.

3.3 Model Details

Model details are divided into two parts: the establishment process, which includes the voting functions model, and the operation process, which includes the task processing model, voting model, and benefit model.

3.3.1 Task Processing Model

For simplicity, we define the first M servers in $\mathcal{S}$ as stable servers. The remaining servers are temporarily connected with a maximum of N servers which the server set $\mathcal{S}$ is represented as $\mathcal{S} = \{k_1, \ldots, k_m, \ldots, k_M, k_{M+1}, \ldots, k_{M+N}\}$. Obviously, a total of $M + N$ servers can be concurrently connected during the specified time horizon.

The number of tasks in each round follows a Poisson distribution: $\mathbb{P}(X = k) = \frac{\lambda^k}{k!} \exp\{-\lambda\}$. Therefore, the number of tasks generated in the t-th round is denoted as λ_t. In the t-th round, the i-th task is denoted as $\omega_{i,t} = \left(\omega_{i,t}^F, \omega_{i,t}^B, \pi_{t,k_m}(i), \omega_{i,t}^{P,1}, \omega_{i,t}^{P,2}\right), i \in \{\lambda_t\}$ where one of $\omega_{i,t}^{P,1}$ or $\omega_{i,t}^{P,2}$ is 0.

The task processing model illustrates transferring tasks from mobile devices to edge servers. We model this process as consuming time and fund costs.

(1) Delay for a single task. A TODMC collects all tasks, and the target server $\pi_{t,k_m}(i)$ for $\omega_{i,t}$ is determined by the algorithm, where $\pi_{t,k_m}(i) \in [\mathcal{S}]$. The total delay for $\omega_{i,t}$ is denoted by $\mathcal{D}(\omega_{i,t})$:

$$\mathcal{D}(\omega_{i,t}) = \mathcal{D}^{up}(\omega_{i,t}) + \mathcal{D}^{exe}(\omega_{i,t}) + \mathcal{D}^{down}(\omega_{i,t}), \quad (1)$$

where $\mathcal{D}^{up}(\omega_{i,t})$ denotes the task upload delay, $\mathcal{D}^{exe}(\omega_{i,t})$ denotes the task execution delay, and $\mathcal{D}^{down}(\omega_{i,t})$ denotes the result download delay:

$$\begin{aligned} \mathcal{D}^{up}(\omega_{i,t}) &= \frac{\omega_{i,t}^F}{r_{\pi_{t,k_m}(i)}}, \mathcal{D}^{down}(\omega_{i,t}) = \frac{\omega_{i,t}^B}{r_{\pi_{t,k_m}(i)}}, \\ \mathcal{D}^{exe}(\omega_{i,t}) &= \frac{\omega_{i,t}^F}{v_{\pi_{t,k_m}(i)}} = \omega_{i,t}^F \cdot \frac{\beta}{f_{\pi_{t,k_m}(i)}}, \end{aligned} \quad (2)$$

And following the Shannon formula, we have

$$r_{\pi_{t,k_m}(i)} = W_{\pi_{t,k_m}(i)} \log_2\left(1 + \frac{p_{\pi_{t,k_m}(i)} h_{\pi_{t,k_m}(i)}}{\mathcal{N}}\right), \quad (3)$$

where $r_{\pi_{t,k_m}(i)}$ denotes the transmission rate between TODMC and the $\pi_{t,k_m}(i)$-th server; $W_{\pi_{t,k_m}(i)}$ is the channel bandwidth to the $\pi_{t,k_m}(i)$-th server, which is fixed but unknown; $p_{\pi_{t,k_m}(i)}$ indicates the power consumption level for uploading tasks to the $\pi_{t,k_m}(i)$-th server; $h_{\pi_{t,k_m}(i)}$ signifies the channel gain between TODMC and the $\pi_{t,k_m}(i)$-th server; $\mathcal{N}$ is random white sub-gaussian noise power; $v_{\pi_{t,k_m}(i)}$ is the CPU capability of the $\pi_{t,k_m}(i)$-th server; β is the average CPU cycles needed to execute one bit of information; and $f_{\pi_{t,k_m}(i)}$ is the CPU clock frequency of the $\pi_{t,k_m}(i)$-th server.

(2) Fund cost for a single task [36]. A fixed fee for a single task is not suitable for edge servers with different computing capacities. Therefore, a dynamic fund cost for a single task includes computing resource consumption and energy consumption. For $\omega_{i,t}$, its fund cost is denoted as $\mathcal{C}(\omega_{i,t})$:

$$\mathcal{C}(\omega_{i,t}) = \mathcal{C}^{res}(\omega_{i,t}) + \mathcal{C}^{eng}(\omega_{i,t}), \quad (4)$$

where $\mathcal{C}^{res}(\omega_{i,t})$ is the computing resources consumption and $\mathcal{C}^{eng}(\omega_{i,t})$ is the energy consumption, which is obtained as follows:

$$\begin{aligned} \mathcal{C}^{res}(\omega_{i,t}) &= a \cdot \omega_{i,t}^F, \\ \mathcal{C}^{eng}(\omega_{i,t}) &= e_1 \cdot \mathcal{D}^{exe}(\omega_{i,t}) + e_2 \cdot \mathcal{D}^{down}(\omega_{i,t}), \end{aligned} \quad (5)$$

where a is the fee-charging rate for processing one-bit information, e_1 is the CPU energy charging standard per second, and e_2 is the signal transmitter energy charging standard per second.

TABLE 2
Definition of Parameters

Notation	Definition
$\mathcal{S}$; $\mathcal{S}_{sta}$; $\mathcal{S}_{tem}$	Server set contains $M+N$ edge server; Stable server set contains M servers $\mathcal{S}_{sta} = \{k_1, \ldots, k_m, \ldots, k_M\}$; Temporary server set contains N servers $\mathcal{S}_{tem} = \{k_{M+1}, \ldots, k_{M+N}\}$.
$\mathcal{T}$; t	Round horizon. Denoted as $\mathcal{T} = \{1, 2, \ldots, t, \ldots, T\}$ where $t \in \mathcal{T}$.
π; $\pi_{t,k_m}(i)$	Task-offloading policy; Target server for task execution.
ω; $\omega_{i,t}$; $\omega_{i,t}^F$; $\omega_{i,t}^B$; $\omega_{i,t}^{P,1}$; $\omega_{i,t}^{P,2}$	A task; Task in t-th round generated by the i-th user; The size of the task; The size of the result; The limited delay of the task; The limited delay of the budget.
$\mathcal{D}(\cdot)$; $\mathcal{D}^{up}(\cdot)$; $\mathcal{D}^{down}(\cdot)$; $\mathcal{D}^{exe}(\cdot)$	Task delay; Task upload delay; Task download delay; Task execution delay.
v; f; β; r; W; h; $\mathcal{N}$	CPU capacity; Average CPU cycles needed to executed one bit of task; CPU clock frequency; Transmission rate; Channel bandwidth; Channel gain; Random white sub-gaussian noise power.
$\mathcal{C}(\cdot)$; $\mathcal{C}^{res}(\cdot)$; $\mathcal{C}^{eng}(\cdot)$; $\mathcal{D}^{exe}(\cdot)$	Fund cost; Computing resource consumption; Energy consumption.
a; e_1; e_2	fee-charging rate for processing one-bit information; CPU energy charging standard per second; Signal transmitter energy charging standard per second.
$u_{k_m}(cdot)$; α_1; α_2; $\hat{D}$; $\hat{C}$; $\mathcal{P}_1$	Voting function; Preference parameters; Delay performance of the policy; Fund cost of the policy; Policy set of one's own.
$\mathcal{R}(\cdot)$; $\{\cdot\}^*$; λ	Regret performance; God policy; Parameter of the Poisson distribution.
Φ_t; $\phi_{i,t}$; $\phi_{i,t}^{id}$; $\phi_{i,t}^{c}$; $\phi_{i,t}^{d}$	Recording matrix; Piece of server information; Connected server ID; Budget cost performance tuple; Delay performance tuple.
$\mathcal{L}_t$; $\mathcal{L}_{t,c}$; $\mathcal{L}_{t,d}$	Idle server set; Idle budget preference server set; Idle delay preference server set.

3.3.2 Voting Function and Voting Model

(1) Voting Function. The purpose of the voting function is to assess the effectiveness of a task-offloading policy. Different committee members have varying voting functions, with each function being inherently self-serving, indicating that the performance of their task-offloading policy is superior under their own voting function. A server with high task processing speed is inclined to undertake more tasks, leading to greater rewards over a stable period. However, this greedy bias is constrained, suggesting that if a particular policy proves to be sufficiently effective, committee members will still vote for it. Let π_{t,k_m} represent the task-offloading policy formulated by the k-th committee member in the t-th round where:

$$\pi_{t,k_m} = \bigcup_{i \in \{\lambda\}} \pi_{t,k_m}(i), \tag{6}$$

The voting function is:

$$\begin{aligned} u_{k_m}(\pi_{t,k_m}) =& \alpha_1 \cdot \left(\frac{\sum_{i \in \{\lambda_t\}} \mathcal{D}(\omega_{i,t}) - \hat{\mathcal{D}}}{\hat{\mathcal{D}}} \right) \\ &+ \alpha_2 \cdot \left(\frac{\sum_{i \in \{\lambda_t\}} \mathcal{C}(\omega_{i,t}) - \hat{\mathcal{C}}}{\hat{\mathcal{C}}} \right), \end{aligned} \tag{7}$$

where $\hat{\mathcal{D}}$ denotes the delay performance of its policy (with id k_m), $\hat{\mathcal{C}}$ is the fund cost of its policy, and α_1, α_2 signify the preference parameters of the k_m-th committee member. The principle behind Eq.(7) is to assess the time and fund costs of other committee members' policies relative to one's own policy, thereby determining their comparative advantages. The ratio of advantages and disadvantages is calculated, with the influence of these factors controlled by the preference parameters α_1 and α_2. The computing resource consumption significantly outweighs energy consumption in the MEC scenario. Hence, we can conveniently ignore the impact of energy consumption disparity and adjust the voting function to the following form:

$$\begin{aligned} u_{k_m}(\pi_{t,k_m}) =& \alpha_1 \cdot \left(\frac{\sum_{i \in \{\lambda\}} \mathcal{D}(\omega_{i,t}) - \hat{\mathcal{D}}}{\hat{\mathcal{D}}} \right) \\ &+ \alpha_2 \cdot \left(\frac{\sum_{i \in \{\lambda\}} \mathcal{C}^{res}(\omega_{i,t}) - \hat{\mathcal{C}}^{res}}{\hat{\mathcal{C}}^{res}} \right). \end{aligned} \tag{8}$$

(2) Voting Model. Denote the committee member set as $\mathcal{X}$. Each member establishes its voting function and a policy performance ranking list, denoted as $\mathcal{V}_{k_m}, k_m \in \mathcal{S}$. After one's policy is made, it broadcasts its policy and receives the others' policies grouping as the policy set $\mathcal{P}_1$:

$$\mathcal{P}_1 = \bigcup_{k_m \in \mathcal{X} \setminus \{1\}} \pi_{t,k_m}, \tag{9}$$

where $\{1\}$ is one's own policy.

Calculate the votes by evaluating all policies, rank them, and select the top one to be broadcast. The policy with the highest votes will be chosen as the next task-offloading policy, and its proponent will become the bookkeeper.

3.4 Problem Formulation

The primary objective of **P1** is to reduce the delay and budget cost of the entire network. Solving **P1** could minimize delays in task processing, enhance the user experience, and increase server profits.

To minimize the sum of both the average delay consumption and the budget cost consumption during the whole time horizon after normalization [1], the optimization problem could be formulated as follows:

1. The $\mathcal{D}(\cdot)$ and $\mathcal{C}(\cdot)$ in **P1** indicate the value after normalization. The normalization method used here is the Min-Max normalization, where the minimum and maximum performance of the system can be borrowed from historical data of other similar networks.

$$\textbf{P1:}\min \frac{\sum_{t=1}^{T}\sum_{i=1}^{\lambda_t}\mathcal{D}\left(\omega_{i,t}\right)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\omega_{i,t}^{F}}+\frac{\sum_{t=1}^{T}\sum_{i=1}^{\lambda_t}\mathcal{C}\left(\omega_{i,t}\right)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\left(\omega_{i,t}^{F}+\omega_{i,t}^{B}\right)}, \tag{10}$$

$$\text{s.t.}\quad \pi_{t,k_m}(i)\in\mathcal{S} \quad \text{(server constraints)} \tag{10a}$$

$$t\in\mathcal{T}, i\in\{\lambda_t\} \quad \text{(round horizon constraints)} \tag{10b}$$

The server constraints (i.e., 10a) indicate that the task-processing server must be selected from $\mathcal{S}$, while the round horizon constraints (i.e., 10b) specify that the network should operate within the defined rounds.

Theorem 1. *P1 is a non-convex problem.*

Proof. According to the server constraints, the $\pi_{t,k_m}(i)\in\mathcal{S}$ where $\mathcal{S}$ is a dynamic set where temporary servers are not settled. The range of values of the independent variable is a non-convex set, so the **P1** is a non-convex problem. □

Firstly, **P1** is a non-convex problem. Moreover, solving **P1** requires detailed knowledge of various critical network state information. As a result, **P1** is challenging to solve. Therefore, we aim to reformulate the problem within the regret framework and bound its regret to find a sub-optimal solution, which is a common approach for addressing similar cold-start optimization problems [32, 37].

The regret framework has three main components: the god policy, the applied algorithm, and the regret. The god policy is an ideal algorithm that minimizes delays and costs, but it's unrealistic due to unpredictable conditions. The applied algorithm is a practical choice based on expertise and considerations. Regret measures the performance gap between the applied algorithm and the god policy, highlighting areas for improvement. By using this framework, we could translate **P1** to **P2**. The following are the regret of delay $\mathcal{R}(\mathcal{D})$ and budget $\mathcal{R}(\mathcal{C})$ of the network:

$$\begin{aligned}\mathcal{R}(\mathcal{D})&=\sum_{t=1}^{T}\sum_{i=1}^{\lambda_t}\left(\mathcal{D}\left(\omega_{i,t}:\{\pi_{t,k_m}(i)\}^{*}\right)-\mathcal{D}\left(\omega_{i,t}\right)\right),\\ \mathcal{R}(\mathcal{C})&=\sum_{t=1}^{T}\sum_{i=1}^{\lambda_t}\left(\mathcal{C}\left(\omega_{i,t}:\{\pi_{t,k_m}(i)\}^{*}\right)-\mathcal{C}\left(\omega_{i,t}\right)\right),\end{aligned} \tag{11}$$

where $\{\pi_{t,k_m}(i)\}^{*}$ is the god policy with the least delay and budget cost. The **P2** tries to minimize the regret of both delay and budget cost.

$$\textbf{P2:}\min\left(\frac{\mathcal{R}(D)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\omega_{i,t}^{F}}+\frac{\mathcal{R}(C)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\left(\omega_{i,t}^{F}+\omega_{i,t}^{B}\right)}\right), \tag{12}$$

$$\text{s.t.}\quad \pi_{t,k_m}(i)\in\mathcal{S} \quad \text{(server constraints)} \tag{12a}$$

$$t\in\mathcal{T}, i\in\{\lambda_t\} \quad \text{(round horizon constraints)} \tag{12b}$$

$$\mathcal{D}(\omega_{i,t})\le\omega_{i,t}^{P,1} \quad \text{(delay constraints)} \tag{12c}$$

$$\mathcal{C}(\omega_{i,t})\le\omega_{i,t}^{P,2} \quad \text{(budget cost constraints)} \tag{12d}$$

The delay constraints (i.e., 12c) and budget cost constraints (i.e., 12d) are the maximum time overhead and budget overhead acceptable for the task.

P2 is an NP-Hard problem. We prove the NP-hardness of **P2** by reducing P2 to the Set Cover Problem (SCP) which is a well-known NP-hard problem [38]. The following is the reduction process.

- **Description of P2.** In the regret framework, task-offloading policies π_{t,k_m} (e.g., subsets in SCP) are selected over rounds t, aiming to minimize regret where the regret is the difference between the chosen policies and the optimal solution.
- **Reduction.** Map SCP to the **P2**. (1) Map each subset as a task-offloading policy π_{t,k_m}. (2) Map the feedback (i.e., delay performance $\mathcal{D}(\omega_{i,t})$ and cost performance $\mathcal{C}(\omega_{i,t})$) of each policy as the number of elements covered. (3) Map the "server constraints" as constraints in SCP that when selecting subsets, each subset must include certain specific elements. (4) Map the "round horizon constraints" as the maximum number of selected subsets. (5) Map the "delay constraints" and "budget cost constraints" as penalties when subsets include specific groups of elements. (6) Map the minimization of regret in **P2** to covering all elements with the fewest subsets.

In addition, two examples are provided to illustrate that a task-offloading policy (made by applying the UCB1 algorithm) cannot be verified as optimal within polynomial time, confirming that P2 is indeed an NP-hard problem.

- **Case 1: Stable Server Space.** Consider a system that contains two policies whose performance is in $[x_1-\tau, x_1+\tau]$, $\left[x_2-\tau^{'}, x_2+\tau^{'}\right]$, where $x_1=x_2$ are the performance empirical mean and $\tau=\sqrt{t/A_1}, \tau^{'}=\sqrt{t/A_2}$ are the confidence radius with policy selected times A_1, A_2. To find the best policy, we should prove that $\tau>\tau^{'}$ or $\tau<\tau^{'}$. However, if $t<\infty$, we only have $|\tau-\tau^{'}|>0$.
- **Case 2: Dynamic Server Space.** It is not feasible to verify whether the currently connected server will outperform all future edge servers. This limitation makes the optimal policy for **P2** inaccessible, thereby confirming that **P2** is an NP-hard problem.

Consequently, based on the definition of an NP-hard problem, we conclude that **P2** is indeed NP-hard. A more effective approach, therefore, is to propose an online exploration algorithm that demonstrates linear regret growth, as illustrated below:

$$\lim_{t\to\infty}\left(\frac{\mathcal{R}(D)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\omega_{i,t}^{F}}+\frac{\mathcal{R}(C)}{\sum\limits_{t\in\mathcal{T},i\in\{\lambda_t\}}\left(\omega_{i,t}^{F}+\omega_{i,t}^{B}\right)}\right)\to 0. \tag{13}$$

4 Design Details of TSAPO

In this section, we present a Thompson Sampling-based Adaptive Preference Optimization algorithm (TSAPO) to address the problem **P2**. The TSAPO algorithm is designed for an individual CVTOM committee member, ensuring that the decisions align with the member's expertise.

4.1 Reflecting the MAB Framework into CVTOM

The MAB framework consists of decision-making problems where an agent follows a round policy, trying different actions (or arms), and receiving rewards or penalties. The challenge lies in balancing exploration of new actions for potential rewards with exploitation of known high-reward actions, aiming to make optimal decisions in uncertain environments to maximize cumulative rewards. The round policy typically follows an algorithm, with Thompson Sampling being one of them. In the CVTOM, the agent is replaced by TODMC, a decentralized and transparent entity, enhancing overall system stability and Security. Each "action" of the bandit is an option edge server in $\mathcal{S}$ that can process computing tasks. The goal shifts from maximizing the cumulative action reward to minimizing cumulative regret as denoted in **P2**.

The MAB framework involves decision-making problems where an agent follows a round-based policy, testing different actions (or arms) and receiving rewards or penalties. The main challenge is balancing the exploration of new actions for potential rewards with the exploitation of known high-reward actions to make optimal decisions in uncertain environments and maximize cumulative rewards. This policy typically uses algorithms such as Thompson Sampling. In most existing MEC scenarios, the agent is a centralized base station that independently makes task-offloading decisions for free communication. However, in the proposed CVTOM, the agent is replaced by TODMC, a decentralized and transparent entity that improves system stability and security. Each "action" in the bandit framework is an edge server in $\mathcal{S}$ capable of processing computing tasks. The goal shifts from maximizing cumulative rewards to minimizing cumulative regret, as defined in P2

Before introducing the TSAPO algorithm, some useful notations were made. Denote g as the prior distribution for setting up Thompson sampling (generally set as Gaussian or sub-gaussian distribution). Denote Φ_t as the recording matrix at the t-th round, $\Phi_t = \{\phi_{1,t}, \phi_{2,t}, \ldots, \phi_{i,t}, \ldots\}^{\mathrm{T}}$ where $\{\cdot\}^{\mathrm{T}}$ is the matrix transpose. The recording matrices record all the essential information of edge servers in $\{\mathcal{S}\}$ stored at each TODMC member. For each piece of server information $\phi_{i,t} = \left(\phi_{i,t}^{\mathrm{id}}, \phi_{i,t}^{c}, \phi_{i,t}^{d}\right)$ where $\phi_{i,t}^{\mathrm{id}}$ is the connected server ID number which satisfy $\phi_{i,t}^{\mathrm{id}} \leq M + N$; $\phi_{i,t}^{c}$ is the budget cost performance tuple; and $\phi_{i,t}^{d}$ is the delay performance tuple. $\phi_{i,t}^{c} = \left(\phi_{i,t}^{c,1}, \phi_{i,t}^{c,2}, \phi_{i,t}^{c,3}, \phi_{i,t}^{c,4}, \phi_{i,t}^{c,5}, \phi_{i,t}^{c,6}\right)$ where $\phi_{i,t}^{c,1}$ is the maximum observation of task execution cost during the first t round, $\phi_{i,t}^{c,2}$ is the minimum observation of task execution cost during the first t round, $\phi_{i,t}^{c,3}$ is the average task execution cost, $\phi_{i,t}^{c,4}$ is the selected times during the first t round, $\phi_{i,t}^{c,5}$ is the mean value of the gaussian conjugate parameter, and $\phi_{i,t}^{c,6}$ is the variance of the gaussian conjugate parameter. $\phi_{i,t}^{d}$ follows the same structure as $\phi_{i,t}^{c}$.

Denote a **sub-optimal server** as an action that still exists in the recording matrix. To be classified as sub-optimal, there are two ways: the performance gap between sub-optimal servers and the observed optimal server is tolerable, or the server is just connected to the CVTOM system.

Algorithm 1 TSAPO Algorithm

Input:
The set of Stable Server and connected Temporary Server; The recording matrix Φ_t; Initialize prior distribution as Gaussian-Gaussian conjugate Priors $\mathcal{N}(\phi_{i,t}^{c,5}, \phi_{i,t}^{c,6}), \mathcal{N}(\phi_{i,t}^{d,5}, \phi_{i,t}^{d,6}), t \in [T], i \in \{\lambda_t\}$;

Output:
A series of task-offloading policy;

1: **for** the first batch **do**
2: Try each server in Φ_t and update the corresponding arguments in recording matrix;
3: **end for**
4: **for** $t = 1 : T$ **do**
5: TODMC collects λ_t task requests;
6: **Run Server Modification algorithm (Algorithm 2)**;
7: Get idle server set $\mathcal{L}_t = \mathcal{L}_{t,c} \cup \mathcal{L}_{t,d}$;
8: **while** True **do**
9: **if** $\omega_{i,t}^{P,1} \neq 0$ and $\|\mathcal{L}_{t,c}\| \geq 2$ **then**
10: Let $\pi_{t,k_m}(i) = \arg_{\phi_{i,t}^{\mathrm{id}} \in \mathcal{L}_{t,c}} \min \mathcal{N}(\phi_{i,t}^{c,5}, \phi_{i,t}^{c,6})$;
11: **else if** $\omega_{i,t}^{P,2} \neq 0$ and $\|\mathcal{L}_{t,d}\| \geq 2$ **then**
12: Let $\pi_{t,k_m}(i) = \arg_{\phi_{i,t}^{\mathrm{id}} \in \mathcal{L}_{t,d}} \min \mathcal{N}(\phi_{i,t}^{d,5}, \phi_{i,t}^{d,6})$;
13: **end if**
14: **if** $\pi_{t,k_m}(i)$ is not None **then**
15: Offload task to $\pi_{t,k_m}(i)$ and update Φ_t;
16: Collect $\pi_{t,k_m}(i)$ into task-offloading policy $\mathcal{P}_1$;
17: **end if**
18: **if** All tasks have been offloaded **then**
19: Break;
20: **end if**
21: **end while**
22: Broadcast $\mathcal{P}_1$ to the other TODMC numbers;
23: **end for**

4.2 Detailed Explanation of TSAPO and Its Advantages

The TSAPO algorithm explanation is divided into three parts: an in-depth introduction of TSAPO, a detailed overview of the Server Modification, and a comparison of its benefits with leading MAB-based algorithms.

Introduction to TSAPO. As shown in Algorithm 1, TSAPO helps a committee member develop task-offloading policies over multiple rounds to win the vote. The input to TSAPO includes key parameters, especially the Gaussian-Gaussian conjugate distribution arguments for sampling server capacity. The output is a sequence of task-offloading policies. In lines 1–3, the algorithm gathers available servers and records their performance metrics, including delay and budget cost. When receiving λ_t task requests in the t-th round, the algorithm identifies the available servers using the Server Modification algorithm (Algorithm 2). In lines 9–13, after classifying the tasks, the Thompson Sampling algorithm is used to sample and choose either the minimum budget cost or delay to process the current task. This iterative process continues until all tasks are assigned to their respective edge servers, forming the final policy for the round. Finally, this policy is broadcast to other TODMC members for voting. The TSAPO algorithm is compared with traditional MAB-based algorithms within the TODMC framework, and the resulting policies are evaluated.

Introduction to the Server Modification Algorithm.

Algorithm 2 Server Modification Algorithm

Input:
The recording martix Φ_t;
Output:
Idle server set $\mathcal{L}_t = \mathcal{L}_{t,c} \cup \mathcal{L}_{t,d}$;

1: Set $\{\mathcal{L}_t\} = \emptyset$;
2: Find the observed optimal server for saving budget cost as
$$c : \phi_{i,t}^{\mathrm{id}} = \arg_{\phi_{i,t}^{\mathrm{id}}} \min \phi_{i,t}^{c,3};$$
3: Find the observed optimal server for saving delay as
$$d : \phi_{i,t}^{\mathrm{id}} = \arg_{\phi_{i,t}^{\mathrm{id}}} \min \phi_{i,t}^{d,3};$$
4: **for** $\phi_{i,t}^{\mathrm{id}} \in \Phi_t \backslash \{c : \phi_{i,t}^{\mathrm{id}}\}$ with no *budget cost None* **do**
5: **if** $\phi_{i,t}^{c,3} \geq \phi_{c:\phi_{i,t}^{\mathrm{id}},t}^{c,3} + \sqrt{\frac{\xi_c \ln \sum_{j=1}^{t} \lambda_j}{\phi_{i,t}^{c,4}}}$ **then**
6: Mark $\phi_{i,t}^{\mathrm{id}}$ as *budget cost None*;
7: **else**
8: Add $\phi_{i,t}^{\mathrm{id}}$ in $\{\mathcal{L}_{t,c}\}$;
9: **end if**
10: **end for**
11: **for** $\phi_{i,t}^{\mathrm{id}} \in \Phi_t \backslash \{d : \phi_{i,t}^{\mathrm{id}}\}$ with no *delay None* **do**
12: **if** $\phi_{i,t}^{d,3} \geq \phi_{d:\phi_{i,t}^{\mathrm{id}},t}^{d,3} + \sqrt{\frac{\xi_d \ln \sum_{j=1}^{t} \lambda_j}{\phi_{i,t}^{d,4}}}$ **then**
13: Mark $\phi_{d,t}^{\mathrm{id}}$ as *delay None*;
14: **else**
15: Add $\phi_{i,t}^{\mathrm{id}}$ in $\mathcal{L}_{t,d}$;
16: **end if**
17: **end for**
18: **if** New servers appears **then**
19: Add it into Φ_t, $\mathcal{L}_{t,c}$ and $\mathcal{L}_{t,d}$;
20: **end if**
21: **if** Servers offline **then**
22: Mark $\phi_{i,t}^{\mathrm{id}}$ as both *budget cost None* and *delay None*;
23: **end if**

The Server Modification Algorithm, outlined in Algorithm 2, has two main objectives. First, it detects and filters out underperforming servers to prevent them from processing certain tasks. Second, it integrates new servers into the network or removes servers that have lost connectivity. The algorithm takes the t-th round recording matrix, Φ_t, as input and outputs two sets of servers: one for powerful servers and another for idle servers.

In line 1, the algorithm initializes the set of idle servers as empty. Lines 2–17 focus on detecting underperforming servers, while lines 18–23 handle the integration and removal of servers. Sub-optimal servers are identified by excluding those marked as "budget cost None" or "delay None". The algorithm then compares these servers with the observed optimal server and filters out the underperforming ones. This is done through the if statements in lines 5 and 12, which follow a policy similar to the UCB algorithm. The UCB approach compares the lower confidence bound of sub-optimal servers with the upper confidence bound of the optimal server. If the upper bound of the optimal server is lower, it means that server's task processing cost or delay is too high and is no longer worth considering. On the other hand, if the upper bound of the optimal server is higher, further exploration is needed. Lines 18 and 21 use if statements to manage servers that are either online or offline

Algorithmic Complexity Analysis. From the pseudo-code of the algorithm, it can be seen that the complexity of the sampling operation for all connectable servers in a round is $\mathcal{O}(M+N)$ and the complexity of the Server Modification Algorithm is $\mathcal{O}(M+N)$. Therefore, the time complexity of the TSAPO is $\mathcal{O}(T(M+N))$

Advantages of the TSAPO. Several algorithms, such as the UCB-based algorithm [19], have been designed to solve MAB problems. In addition, there are classic algorithms such as Epsilon-Greedy, Exploit-then-Commit (ETC), etc. However, these algorithms are not suitable for solving exploration and exploitation (EE) dilemmas in CVTOM environments. The main differences in the environments and the advantages are summarized as follows:

1) The agent (TODMC) could pull multiple arms (servers) in a single round.
2) Due to the existence of migration servers, the exact number of arms remains unconfirmed, causing changes in the recording matrix.
3) The task preferences are different and the optimization variables are not single.

The regret upper bound of the TSAPO according to Eq. ((12)) is as follows.

Theorem 2. *According to* ***P2****, if all tasks collected are delay-sensitive or budget-sensitive, the regret of the MEC system by using TSAPO is finally bounded by the following, which is sub-linear:*

$$\lim_{T\to\infty}\left(\frac{\mathcal{R}}{\sum_{t\in T, i\in\lambda_t}\omega_{i,t}^{F}(\omega_{i,t})}\right) \propto \frac{\sqrt{\lambda T(M+N)\log(\lambda T)}}{T}, \tag{14}$$

where T is the amount of the looping times; λT is the number of tasks collected in CVTOM; M is the number of stable servers; and N is the amount of connected temporary servers throughout the entire round horizon.

Lipschitz Analysis*. Let a random T real function $\mathcal{G}: \mathcal{R} \to \mathcal{R}$. If Eq. (14) is proven, it means that the growth rate of regret is sub-linear with time. According to the setting of Fig. 7, by setting $M+N=20$ and $T>40$ would satisfy the Lipschitz condition.*

Proof. The proof sketch. (1) Initially, preliminary work is conducted, comprising modeling CVTOM as a Bayesian bandit environment, the normalization of regret, the definition of the two opposing events $\mathcal{H}$ and $\mathcal{H}^{'}$, and the definition of a useful function $\mathcal{K}_t(i)$. (2) Subsequently, by using $\mathcal{K}_t(i)$, $\mathcal{H}$ and $\mathcal{H}^{'}$, Lemma 4.1 is proven. According to Lemma 4.1, the processing servers have an acceptable capacity under the TSAPO algorithm. (3) Finally, under the task number assumption and Lemma 4.1, we can prove that Eq. (23) holds (i.e., Theorem 2 holds), which is the total regret of the system by applying TSAPO is sublinear. The important conclusions are listed as follows and the lengthy mathematical proof is left in Appendix-A.

We prove Theorem 2 through the utilization of delay regret. The CVTOM system operates within a Bayesian bandit environment $(\mathcal{E}, \mathcal{B}(\mathcal{E}), \mathcal{Q}, \mathcal{P})$ by treating edge servers as arms. Here, $(\mathcal{E}, \mathcal{B}(\mathcal{E}))$ denotes the measurable space; $\mathcal{B}(\mathcal{E})$ signifies a Borel space within $\mathcal{E}$; $\mathcal{Q}$ reflects an arbitrary prior conjugate distribution (specifically Gaussian-Gaussian in this paper) defined on the aforementioned measurable

space; and $\mathcal{P}$ is the corresponding kernel probability. Let $\mathcal{F}_t = \sigma(A_1, X_1, \ldots, A_t, X_t)$ be the σ-algebra generated by the first t rounds of interactions. Set $n = T$ as the time horizon. To demonstrate the progress of the proof, we make the following preparations and assumptions.

Preparations Normalize the regret into $[0, 1]$**.** Denote the delay performance gap as:

$$\Delta_d = \max_{i \in \mathcal{S}, t \in \mathcal{T}} \phi_{i,t}^{d,3} - \min_{i \in \mathcal{S}, t \in \mathcal{T}} \phi_{i,t}^{d,3}. \tag{15}$$

Then the normalized delay bonus $r_i(t)$ in the t-th round can be denoted as (abbreviated as r_i):

$$r_i(t) = \frac{\max_{j \in \mathcal{S}, t' \in \mathcal{T}} \phi_{j,t'}^{d,3} - \phi_{i,t}^{d,3}}{\Delta_d}. \tag{16}$$

And we assume the best server is:

$$\mathcal{A}^* = \arg\max_{i \in \mathcal{S}} r_i(t). \tag{17}$$

Define two events. Denote the correct estimation $\mathcal{H}, \forall t \in n, i \in \mathcal{S}$ and incorrect estimation $\mathcal{H}^{'}, \exists t \in n, i \in \mathcal{S}$:

$$\begin{aligned} \mathcal{H} &= \left\{ |\hat{r}_i(t-1) - r_i| < \sqrt{\frac{2\log(1/\delta)}{\max\left(1, \phi_{i,t-1}^{d,4}\right)}} \right\}, \\ \mathcal{H}^{'} &= \left\{ |\hat{r}_i(t-1) - r_i| \geqslant \sqrt{\frac{2\log(1/\delta)}{\max\left(1, \phi_{i,t-1}^{d,4}\right)}} \right\}, \end{aligned} \tag{18}$$

where $\hat{r}_i(t-1)$ is the delay bonus empirical mean of the i-th server after $t-1$ rounds of interaction with the environment. Here, $\delta \in [0, 1]$, typically defined as $1/n^2$. It is specified that if $\phi_{i,t-1}^{d,4} = 0$, then $\hat{r}_i(t-1) = 0$.

Define clip function. Define the function $\mathcal{K}_t(i)$ as:

$$\mathcal{K}_t(i) = clip_{[0,1]}\left(\hat{r}_i(t-1) + \sqrt{\frac{2\log(1/\delta)}{\max\left(1, \phi_{i,t-1}^{d,4}\right)}} \right), \tag{19}$$

where $clip_{[0,1]}(x) = \max(0, \min(1, x))$.

Task number assumption. For the first t rounds, we assume that:

$$\lim_{t \to \infty} \mathbb{P}\{\sum_{j=1}^{t} \lambda_j = t \cdot \lambda\} \to 1. \tag{20}$$

Under such an assumption, we consider the number of tasks collected to be fixed.

Lemma 4.1. *The performance of servers in $\mathcal{L}_t$ is acceptable.*

Proof. Using **Hoeffding's Inequality**, after the t-th round, the remaining servers are likely to be suboptimal, meaning their performance is still good.

$$\begin{aligned} \mathbb{P}\{\phi_{i,t}^{d,3} - \phi_{\mathcal{A}^*,t}^{d,3} \leq \epsilon\} &\geq 1 - 2\exp\left\{ -\frac{2\epsilon^2 t^2}{\sum_t \left(\phi_{i,t}^{d,1} - \phi_{i,t}^{d,2}\right)^2} \right\} \\ &\geq 1 - 2\exp\left\{ -\frac{2\epsilon^2 t^2}{t\Delta_d^2} \right\} \geq 1 - 2\exp\left\{ -\frac{2\ln\{t\lambda\} t^2 \Delta_d^2}{2t^2\Delta_d^2} \right\} \\ &\geq 1 - 2\exp\{-\ln\{t\lambda\}\} = 1 - \frac{2}{t \cdot \lambda}, \end{aligned} \tag{21}$$

where $\epsilon = \Delta_d \cdot \sqrt{\frac{\ln\{t \cdot \lambda\}}{2t}}$. Therefore, according to the Eq. (21), we have:

$$\lim_{t \to \infty} \mathbb{P}\left\{ \phi_{i,t}^{d,3} - \phi_{\mathcal{A}^*,t}^{d,3} \leq \epsilon \right\} \to 1, \epsilon \to 0, \tag{22}$$

Therefore, we assume that the servers in $\mathcal{L}_t$ are acceptable. □

According to Lemma 4.1, the servers selected at line 12 in Algorithm 1 after t rounds are acceptable. Let $\mathcal{A}_t$ be the server selected by TSAPO and the total regret $\mathcal{R}$ can be formulated as ($\delta = \frac{1}{\lambda^2 T^2}$) and the main proof process is available in the Appendix-A.

$$\mathcal{R} = \mathbb{E}\left[\sum_{t=1}^{n} \left(r_{\mathcal{A}^*} - \mathbb{E}\left[\mathcal{K}_t(\mathcal{A}^*)\right] + \mathbb{E}\left[\mathcal{K}_t(\mathcal{A}^*)\right] - r_{\mathcal{A}_t} \right) \right] \leq \sqrt{32n(M+N)\log(1/\delta)} + 4n^2(M+N)\delta. \tag{23}$$

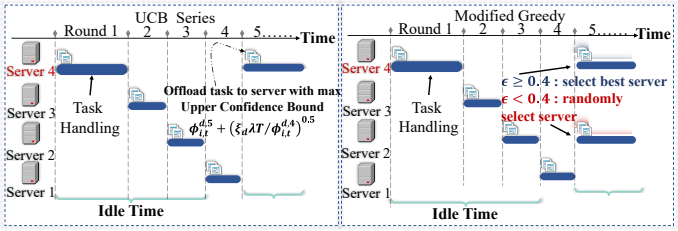


Fig. 3. Tasks Processing Schedule in Traditional MAB Based Algorithms.

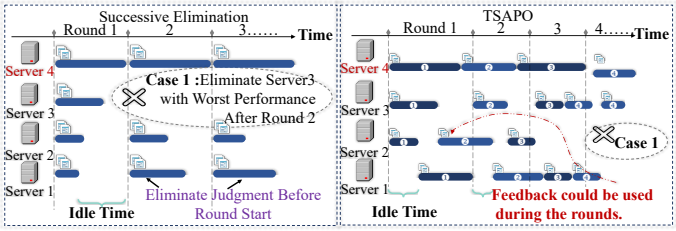


Fig. 4. Tasks Processing Schedule in Parallel Based Algorithms.

5 Simulation Experiments

In this section, a smart contract consumption experiment is first established. Then the CVTOM parameters are set up. Subsequently, the algorithms employed for comparison and the dimensions for evaluating their performance are introduced. Finally, six sets of experiments are listed, and the results are elucidated.

5.1 The Implementation of Smart Contract

Inspired by Wang *et al.* work [39], this experiment is conducted on Ethernet in order to verify the gas cost of the voting mechanism. As Fig. 5 shows, four committee members were given the authority to vote. A smart contract is designed to vote for the best performance task-offloading policy in each round. The test platform is Etherscan and the

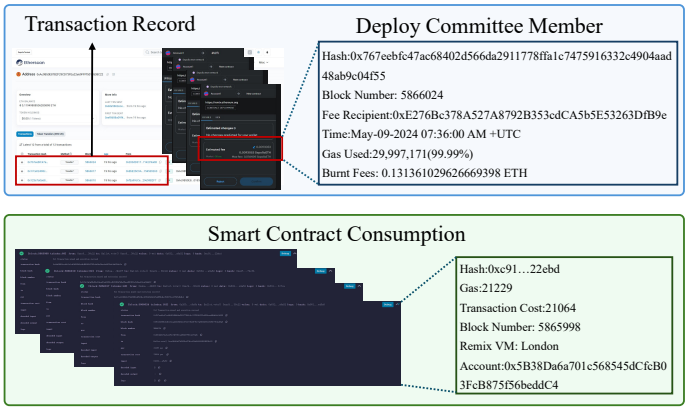


Fig. 5. Smart Contract performance. Its hash address is 0xAc9B50E87B2F29CD73FEa22e69FFFf5018538C22 and the access address is https://sepolia.etherscan.io/address/0xac9b50e87b2f29cd73fea22e69ffff5018538c22. Part of the details are shown in this figure.

test environment is Remix VM (London). The voting performance, e.g., average gas cost, transaction cost, and latency for the execution of the smart contracts are 21229, 21064, and 1ms, respectively. The state of the blockchain network, e.g., average block size, and block reward is 152,888 bytes and 0.068985570191692329 ETH respectively.

5.2 The CVTOM Parameters Setting

Consider a CVTOM consisting of 5 stable servers (also denoted as committee members) and 15 temporary servers. Each stable server has voting function parameters α_1, α_2 randomly selected from the range 0.4 to 0.6. The energy charging standard e_1, e_2 are denoted as 1. Following the voting function, committee members assess the performance of various task-offloading policies and vote for the optimal one. The task processing capacity of these servers ranges from 7.5MB/s to 15MB/s. For TSAPO, the prior conjugate distribution is set as Gaussian-Gaussian with a predictable variance based on empirical data, while the mean value remains unknown. The control parameters of TSAPO, ξ_c, ξ_d, are typically set as 1 in most experiments, except for specific testing scenarios in Fig. 12. Referring to [40], the CPU frequency of servers ranges from 3GHz to 5GHz, and the CPU cycle required to handle a one-bit task is set to 1000. The system's channel bandwidth varies from 15kHz to 30kHz, and the channel gain ranges from 0.625 to 0.875 [41]. The cost of computing resources stands between \$0.02 and \$0.03 [36]. The signal noise power is fixed at 50W.

Consider tasks generated in each round following a Poisson distribution with a parameter λ of 20. Their size varies randomly between 60MB and 100MB. Tasks are generated under two models. The Delay model includes all tasks that are sensitive to delays. The Mix model includes tasks that are delay-sensitive as well as tasks that are budget-sensitive.

5.3 Evaluation Dimensions & Baseline Algorithms

Under the CVTOM system, we evaluate the performance of the TSAPO algorithm and compare the results with modified algorithms from existing studies across different dimensions:

1) **Win Rate**. It shows the likelihood of a policy created by a specific algorithm being the winner as shown in Fig. 1. This probability is influenced by various parameters of voting functions.
2) **Optimal Rate** [15, 32]. It denotes the ratio of optimal server selection to total servers. The higher, the better. In Fig. 11, the optimal rate aligns with the rate in each round, while the others represent the rate across the entire time horizon.
3) **Total Delay** [24, 26]. It makes decisions based on the delay in executing all tasks. The lower, the better.
4) **Total Budget Cost** [36]. It makes decisions based on the task execution budget of all collected tasks. The lower, the better.
5) **Task Dropping Rate**. It is based on the proportion of tasks facing higher execution delays or budget deadlines. It also correlates with the severity of the penalty. The lower, the better.
6) **Idle Rate.** It denotes the ratio of working time to idle time of an edge server before it is disconnected. The idle rate level reflects the algorithm's resource utilization within the system.

The baselines are MAB-based algorithms including UCB Series, Greedy Modified, and Successive Elimination, and non MAB-based algorithm, Genetic Algorithm (GA).

UCB Series is an adaptation of UCB1, such as C-UCB, D-UCB [17] which prioritize previous or recent performance, and AW-CUCB [18]. As shown in Fig. 3, the UCB Series algorithm selects the server with the highest upper confidence bound for processing after the initial four rounds of exploration.

Greedy Modified is based on biased greed, with examples like S-OAMC [42] and truthful greedy [43] utilizing the greedy principle for resource offloading. As shown in Fig. 3, the algorithm relies on ϵ-Greedy. If $\epsilon \geq 0.4$, designate the target server using the UCB1 algorithm. Otherwise, select a server randomly as the target. Traditional MAB-based algorithms, due to lack of parallelism, could handle only one task per round, dropping the rest. This method is inefficient, leading to a high rate of task drops and server idle time.

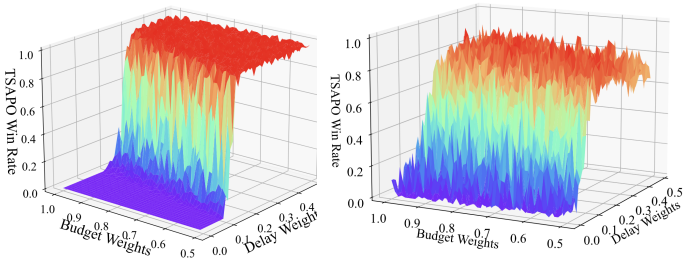
In Fig. 4 (left), the elimination principle aligns with the upper confidence bound of each server, akin to the Server Modification algorithm. Nevertheless, Successive Elimination still encounters prolonged server idle time with an increasing number of servers, potentially assigning tasks repeatedly to underperforming servers.

For each GA experiment, the natural selection is based on whether individual genies contain a faster task-offloading policy. Each genie is the task-offloading times for the corresponding edge server. There is no new server added, the mutation probability is 0.01 and the round is the entire process of GA including selection, crossover, and mutation.

5.4 Experiment Results

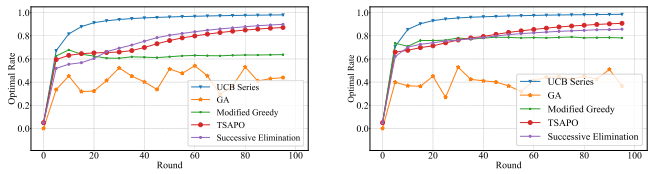
The numerical experiments consist of eight parts. The first experiment, depicted in Fig. 6, aims to validate the probability of the TSAPO algorithm winning the voting across various models. The second experiment, illustrated in Fig. 7, demonstrates the speed of identifying the optimal server

This article has been accepted for publication in IEEE Transactions on Mobile Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TMC.2024.3507153



(a) Delay Model (b) Mix Delay

Fig. 6. TSAPO Win Rate Performance in Different Task Generating Models.



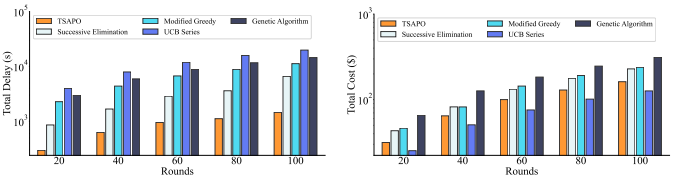
(a) Delay Model (b) Mix Model

Fig. 7. TSAPO Optimal Rate Performance in Different Task Generating Models.

under the strong assumption that the algorithm always wins. The third and fourth experiments, depicted in Fig. 8 and Fig. 9, assess the computational resources and cost-effectiveness of TSAPO. The fifth experiment, shown in Fig. 10, seeks to evaluate the cost efficiency of TSAPO in diverse models. The sixth experiment, displayed in Fig. 11, aims to confirm that TSAPO can effectively balance exploration and exploitation for newly incorporated servers in the system. The seventh experiments, displayed in Fig. 12 aim to ascertain the validity of setting ξ_d, ξ_c to 1. The final experiment, displayed in Fig. 13, seeks to illustrate the utilization of computational resources within the system by each algorithm.

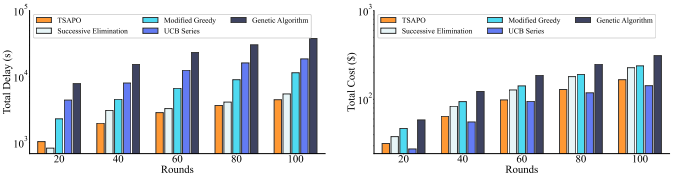
Fig. 6 depicts the Win Rate performance of the TSAPO algorithm compared to other algorithms. Budget Weights are randomly sampled within $[1, 0.5]$, while Delay Weights are randomly sampled within $[0, 0.5]$ for each committee member. The TSAPO algorithm's remarkable capacity for parallel task processing allows it to excel in time efficiency. The win rate of the TSAPO algorithm notably rises, especially in the Delay Model, with an increase in the delay weight within the voting function.

Fig. 7 illustrates the Optimal Rate performance of the algorithms. While the TSAPO doesn't demonstrate a favorable optimal rate performance compared to the UCB Series, the cost of utilizing UCB Series is significantly higher, nearly 1000% more than the TSAPO. The TSAPO selects edge servers for each task only after receiving feedback, and the entire task allocation process is sequential. Consequently, UCB Series, Modified Greedy, and GA result in a notably high total delay, as depicted in Fig. 8 and Fig. 9. In contrast, the TSAPO achieves the shortest task processing duration due to its parallel task allocation and ability to promptly adjust parameters in comparison to the successive elimination algorithm. The number of times the successive elimination algorithm selects the optimal server is lower than that of the



(a) Delay Performance (b) Budget Performance

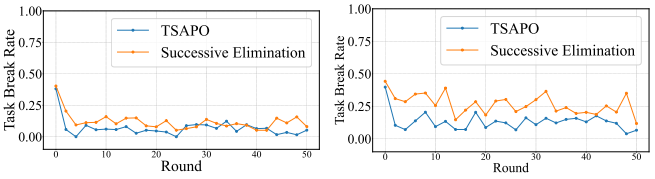
Fig. 8. TSAPO Resources Saving Performance in the Delay Model.



(a) Delay Performance (b) Budget Performance

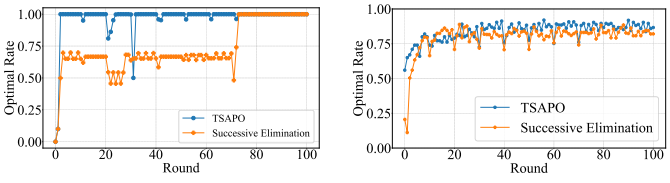
Fig. 9. TSAPO Resources Saving Performance in the Mix Model.

TSAPO because it only processes task feedback at the end of each round.



(a) Delay Model (b) Mix Model

Fig. 10. TSAPO Pentaly Performance.

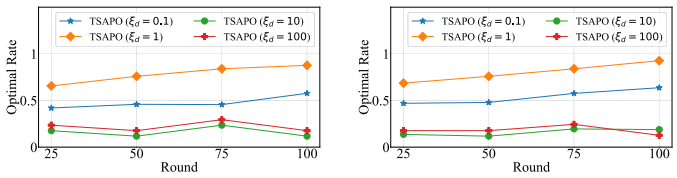


(a) Delay Model (b) Mix Model

Fig. 11. TSAPO Performance Under Server Online/Offline Environment.

Moreover, as depicted in Fig. 8 and Fig. 9, the TSAPO demonstrates acceptable cost-saving capabilities in contrast to the UCB Series at 100 rounds, with approximately 114% and 115% in both the Delay Model and Mix Model. Furthermore, in the Mix Model, the TSAPO outperforms the Successive Elimination approach and non-Mab-based GA algorithms in delay performance beyond 60 rounds.

The purpose of Fig. 10 is to evaluate the task-dropping rate with TSAPO and successive elimination. Results from the Delay model indicate a more stable performance with a lower task-dropping rate compared to the Mix model. TSAPO performs better on both fronts as the least efficient servers remain idle. Fig. 11 aims to assess TSAPO's stability with online server addition and temporary server offline scenarios. In the first 70 rounds, a new temporary server is added every 10 rounds, with a one percent chance of a temporary server offline in subsequent rounds. Results from the Delay model show greater stability



(a) Delay Model (b) Cost Model

Fig. 12. TSAPO Performance Under Different ξ_d, ξ_c setting.

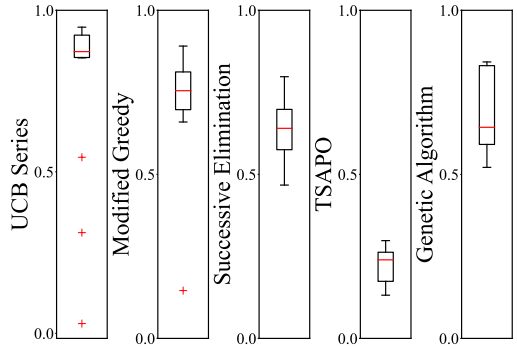


Fig. 13. TSAPO Idle Rate Performance.

than the Mix model, with the rate-down trend attributed to increased server quantity.

Fig. 12 illustrates that setting ξ_d and ξ_c to 1 yields the best performance compared to other values. We tested this in both the delay model and the cost model. The reason for this outcome is that if ξ_d and ξ_c are set to higher values, the equations in Alg. 2, lines 5 and 12, will produce a wider confidence bound, preventing any servers from being eliminated. This would result in behavior similar to a random algorithm. In contrast, if ξ_d and ξ_c are set too low, the selection frequency of the remaining servers with no "delay None" or "budget None" would have negligible impact on the confidence bound of their performance. This would lead to rapid convergence of the delay or cost estimation performance, causing insufficient exploration and potentially eliminating the best-performing server.

Fig. 13 illustrates the utilization of computational resources within the system by each algorithm. The environment for Fig. 13 is set up such that all servers within the environment are stable servers and the case of temporary server connections is not considered. The number of rounds is 100. According to Fig. 13, TSAPO has the best performance with the lowest idle rate.

6 Conclusion

To fully utilize computing resources and reduce task execution delay and budget costs for various sensitive applications such as augmented reality or virtual reality, we introduced the innovative system, CVTOM. In this system, where server numbers were uncertain and key factors unpredictable, we adopted a model inspired by the consortium blockchain network principle. A group of servers formed a committee to collectively train and vote on task-offloading policies, with all task execution data recorded to mitigate risks from malicious servers. We then transformed the core issue into the regret model. Our proposed solution, TSAPO, aimed to optimize voting outcomes and minimize multiple objectives. We also established the upper bound of its regret through rigorous mathematical analysis. Simulation experiments demonstrated that TSAPO, compared to traditional MAB-based algorithms, significantly improved the optimal rate and reduced task execution delays by utilizing a parallel task allocation policy and promptly processing task feedback information.

In future work, we will address the problem of prioritization between tasks. We will explore the issue of contention among users for edge server usage and suggest Nash equilibrium policies to enhance the existing algorithm.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China (No. 62472163, No. 62272154), the Hunan Provincial Natural Science Foundation of China (No. 2024JJ5096), the science and technology innovation Program of Hunan Province (No. 2023RC3125), the Science & Technology talents lifting project of Hunan Province (No. 2023TJ–N23), the Training Program for Excellent Young Innovators of Changsha (No. kq2209008), and the Hunan Provincial Innovation Foundation for Postgraduate (No. CX20230389).

References

[1] K. Namiki, T. Mori, Y. Miyake, and et.al., "Advanced real-time hierarchical task network: Long-term behavior in real-time games," in *Proceedings of the 17th Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2021, pp. 208–212.

[2] A. J. Ferrer, J. M. Marquès, and J. Jorba, "Towards the decentralised cloud: Survey on approaches and challenges for mobile, ad hoc, and edge computing," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 111:1–111:36, 2019.

[3] U. Mohammad, S. Sorour, and M. Hefeida, "Dynamic task allocation for mobile edge learning," *IEEE Trans. Mob. Comput.*, vol. 22, no. 12, pp. 6860–6873, 2023.

[4] H. Wang, H. Xu, H. Huang, and et.al., "Robust task offloading in dynamic edge computing," *IEEE Trans. Mob. Comput.*, vol. 22, no. 1, pp. 500–514, 2023.

[5] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Testing and evaluating a security-aware pub and sub protocol in a fog-driven iot environment," in *19th International Conference on Ad-Hoc Networks and Wireless*, vol. 12338, 2020, pp. 183–197.

[6] Q. Zhang, Z. Zhang, J. Cui, and et.al., "Efficient blockchain-based data integrity auditing for multi-copy in decentralized storage," *IEEE Trans. Parallel Distributed Syst.*, vol. 34, no. 12, pp. 3162–3173, 2023.

[7] Y. Guo, Z. Wan, H. Cui, and et.al., "Vehicloak: A blockchain-enabled privacy-preserving payment scheme for location-based vehicular services," *IEEE Trans. Mob. Comput.*, vol. 22, no. 11, pp. 6830–6842, 2023.

[8] D. C. Nguyen, M. Ding, P. N. Pathirana, and et.al., "Cooperative task offloading and block mining in blockchain-based edge computing with multi-agent deep reinforcement learning," *IEEE Trans. Mob. Comput.*, vol. 22, no. 4, pp. 2021–2037, 2023.

[9] L. Jun, S. Yumeng, W. Kang, and et.al., "Blockchain assisted decentralized federated learning (blade-fl): Performance analysis and resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 10, pp. 2401–2415, 2022.

[10] S. Chen, H. Mi, J. Ping, and et.al., "A blockchain consensus mechanism that uses proof of solution to optimize energy dispatch and trading," *Nature Energy*, vol. 7, no. 6, pp. 495–502, 2022.

[11] H. Jia, C. Jiang, L. Kuang, and J. Lu, "Adaptive access control and resource allocation for random access in NGSO satellite networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 4, pp. 2721–2733, 2022.

[12] T. Ouyang, X. Chen, Z. Zhou, and et.al., "Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning," *IEEE Trans. Mob. Comput.*, vol. 22, no. 3, pp. 1313–1326, 2023.

[13] G. Gao, S. Huang, H. Huang, and et.al., "Combination of auction theory and multi-armed bandits: Model, algorithm, and application," *IEEE Trans. Mob. Comput.*, vol. 22, no. 11, pp. 6343–6357, 2023.

[14] J. Hong, B. Kveton, M. Zaheer, and et.al.h, "Multi-task off-policy learning from bandit feedback," in *International Conference on Machine Learning*, vol. 202, 2023, pp. 13 157–13 173.

[15] Y. Zhu, D. J. Foster, J. Langford, and et.al., "Contextual bandits with large action spaces: Made practical," in *International Conference on Machine Learning*, vol. 162, 2022, pp. 27 428–27 453.

[16] O. Elishco and A. Barg, "Capacity of dynamical storage systems," *IEEE Trans. Inf. Theory*, vol. 67, no. 1, pp. 329–346, 2021.

[17] A. Garivier and E. Moulines, "On upper-confidence bound policies for switching bandit problems," in *Algorithmic Learning Theory - 22nd International Conference, Espoo, Finland*, vol. 6925, 2011, pp. 174–188.

[18] L. Li, J. Shen, B. Wu, and et.al., "Adaptive data placement in multi-cloud storage: A non-stationary combinatorial bandit approach," *IEEE Trans. Parallel Distributed Syst.*, vol. 34, no. 11, pp. 2843–2859, 2023.

[19] J. Tianyuan, T. Jing, X. Pan, and et.al., "Almost optimal anytime algorithm for batched multi-armed bandits," in *21th International Conference on Machine Learning*, 2021, pp. 5065–5073.

[20] X. Xiaolong, Z. Xuyun, G. Honghao, and et.al., "Become: Blockchain-enabled computation offloading for iot in mobile edge computing," *IEEE Trans. on Ind. Inf.*, vol. 16, no. 6, pp. 4187–4195, 2020.

[21] Y. Li, Z. Lin, W. Zhang, and et.al., "Mobile edge computing-enabled blockchain: contract-guided computation offloading," *J. Supercomput.*, vol. 79, no. 7, pp. 7970–7996, 2023.

[22] J. Shi, J. Du, Y. Shen, and et.al., "Drl-based V2V computation offloading for blockchain-enabled vehicular networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 7, pp. 3882–3897, 2023.

[23] S. Guo, Y. Dai, S. Guo, and et.al., "Blockchain meets edge computing: Stackelberg game and double auction based task offloading for mobile blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 5, pp. 5549–5561, 2020.

[24] Z. He, K. Li, and K. Li, "Cost-efficient server configuration and placement for mobile edge computing," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 9, pp. 2198–2212, 2022.

[25] H. Li, S. Ma, T. Wang, and et.al., "HASP: hierarchical asynchronous parallelism for multi-nn tasks," *IEEE Trans. Computers*, vol. 73, no. 2, pp. 366–379, 2024.

[26] X. Zhu and M. Zhou, "Multiobjective optimized deployment of edge-enabled wireless visual sensor networks for target coverage," *IEEE Internet Things J.*, vol. 10, no. 17, pp. 15 325–15 337, 2023.

[27] C. Dong, J. Weng, J. Liu, and et.al., "Fusion: Efficient and secure inference resilient to malicious servers," in *30th Annual Network and Distributed System Security Symposium*, 2023.

[28] Z. Ghodsi, T. Gu, and S. Garg, "Safetynets: Verifiable execution of deep neural networks on an untrusted cloud," in *Annual Conference on Neural Information Processing Systems*, 2017, pp. 4672–4681.

[29] L. Zhao, Q. Wang, C. Wang, and et.al.g, "Veriml: Enabling integrity assurances and fair payments for machine learning as a service," *IEEE Trans. Parallel Distributed Syst.*, vol. 32, no. 10, pp. 2524–2540, 2021.

[30] Y. Xu, Z. Peng, N. Song, Y. Qiu, C. Zhang, and Y. Zhang, "Joint optimization of service caching and task offloading for customer application in mec: A hybrid sac scheme," *IEEE Transactions on Consumer Electronics*, 2024.

[31] L. Wang, Z. Yu, D. Zhang, and et.al., "Heterogeneous multi-task assignment in mobile crowdsensing using spatiotemporal correlation," *IEEE Trans. Mob. Comput.*, vol. 18, no. 1, pp. 84–97, 2019.

[32] A. Slivkins, "Introduction to multi-armed bandits," *CoRR*, vol. abs/1904.07272, 2019. [Online]. Available: http://arxiv.org/abs/1904.07272

[33] X. Wang, J. Ye, and J. C. S. Lui, "Decentralized task offloading in edge computing: A multi-user multi-armed bandit approach," in *2022 IEEE Conference on Computer Communications, London, UK*, 2022, pp. 1199–1208.

[34] M. D. Hossain, T. Sultana, S. Akhter, and et.al., "Computation offloading strategy based on multi-armed bandit learning in microservice-enabled vehicular edge computing networks," in *2023 International Conference on Information Networking*, 2023, pp. 769–774.

[35] Z. Zhu, T. Liu, Y. Yang, and X. Luo, "BLOT: bandit learning-based offloading of tasks in fog-enabled networks," *IEEE Trans. Parallel Distributed Syst.*, vol. 30, no. 12, pp. 2636–2649, 2019.

[36] J. Li, W. Liang, W. Xu, and et.al., "Budget-aware user satisfaction maximization on service provisioning in mobile edge computing," *IEEE Trans. Mob. Comput.*, vol. 22, no. 12, pp. 7057–7069, 2023.

[37] A. Karbasi, V. Mirrokni, and M. Shadravan, "Parallelizing thompson sampling," *Advances in Neu. Inf. Processing Syst.*, vol. 34, pp. 10 535–10 548, 2021.

[38] D. B. Czerbo, "Handbook of theoretical computer science : J. van leeuwen, ed., vol. A: algorithms and complexity, vol. B: formal methods and semantics (elsevier, amsterdam, 1990), 2296 pp., hardcover, dfl. 555.00," *Artif. Intell. Medicine*, vol. 4, no. 4, p. 309, 1992.

[39] X. Wang, Y. Zhao, C. Qiu, and et.al., "Infedge: A

blockchain-based incentive mechanism in hierarchical federated learning for end-edge-cloud communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3325–3342, 2022.

[40] X. Gao, J. Wang, X. Huang, and et.al., "Energy-constrained online scheduling for satellite-terrestrial integrated networks," *IEEE Trans. Mob. Comput.*, vol. 22, no. 4, pp. 2163–2176, 2023.

[41] Z. Xu, G. Xu, H. Wang, and et.al., "Enabling streaming analytics in satellite edge computing via timely evaluation of big data queries," *IEEE Trans. Parallel Distributed Syst.*, vol. 35, no. 1, pp. 105–122, 2024.

[42] E. F. Maleki, L. Mashayekhy, and S. M. Nabavinejad, "Mobility-aware computation offloading in edge computing using machine learning," *IEEE Trans. Mob. Comput.*, vol. 22, no. 1, pp. 328–340, 2023.

[43] M. M. Nejad, L. Mashayekhy, and D. Grosu, "Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds," *IEEE Trans. Parallel Distributed Syst.*, vol. 26, no. 2, pp. 594–603, 2015.

Yang Xu received the Ph.D. degree in Computer Science and Technology from Central South University, China in 2019. From 2015 to 2017, he was a visiting scholar in the Department of Computer Science and Engineering at Texas A&M University, USA. He is currently an Associate Professor with the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include distributed computing, trustworthy computing, blockchain, and federated learning. He has published over 50 articles in international conferences and journals, including IEEE INFOCOM, TSC, TITS, TII, etc. He has served as the General Co-Chair for UbiSec 2024, the Steering Committee Co-Chair for IWCSS 2022, and an active reviewer for over 20 international journal/conference proceedings.

Hangfan Li received his B.S. degree in Internet of Things (IoT) from Hebei University of Technology, Tianjin, China, in 2019. From October 2021 to January 2023, he served as a visiting student at Tsinghua University, Shenzhen, China. He is pursuing a Ph.D. degree in the College of Computer Science and Electronic Engineering at Hunan University, Changsha, China. His research interests include Mobile Edge Computing, Multi-armed Bandits, Blockchain, and Computer Network Security.

Cheng Zhang received his B.S. degree in computer science and technology from Shenyang University of Technology, China, in 2017, and the M.S. degree from Central South University, China, in 2021. He is currently pursuing the Ph.D. degree in the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests mainly focus on the Blockchain & smart contract, and Computer Network Security.

Zhiqing Tang received the B.S. from School of Communication and Information Engineering, University of Electronic Science and Technology of China, in 2015 and the Ph.D. from Department of Computer Science and Engineering, Shanghai Jiao Tong University, China, in 2022. He is currently an Assistant Professor with the Institute of Artificial Intelligence and Future Networks, Beijing Normal University, China. His current research interests include edge computing, resource scheduling, and reinforcement learning.

Xiaoxiong Zhong obtained his Ph.D. degree in Computer Science and Technology from Harbin Institute of Technology, China, in 2015. Following that, he served as a Postdoctoral Research Fellow at Tsinghua University, China, from 2016 to 2018. Currently, he holds the position of associate professor at Peng Cheng Laboratory in Shenzhen, China. His research focuses on network protocol design and analysis, data transmission and analysis in Internet of Thing, and Multi-access Computing.

Ju Ren (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in computer science from Central South University, China, in 2009, 2012, and 2016, respectively. Currently, he is an Associate Professor with the Department of Computer Science and Technology, Tsinghua University, China. His research interests include the Internet of Things, edge computing, edge intelligence, and security and privacy.

Hongbo Jiang (Senior Member, IEEE)received the Ph.D. degree from Case Western Reserve University, in 2008. He is currently a full professor with the College of Computer Science and Electronic Engineering, Hunan University. He is an elected fellow of IET (The Institution of Engineering and Technology), a fellow of BCS (The British Computer Society), senior member of ACM, and full member of IFIP TC6 WG6.2. Now his research focuses on computer networking, especially, wireless networks, data science in Internet of Things, and mobile computing.

Yaoxue Zhang (Senior Member, IEEE) received the B.Sc. degree from the Northwest Institute of Telecommunication Engineering, Xi'an, China, in 1982, and the Ph.D. degree in computer networking from Tohoku University, Sendai, Japan, in 1989. He is currently a Professor with the Department of Computer Science and Technology, Tsinghua University, China. He has published more than 200 papers on peer-reviewed IEEE/ACM journals and conferences. His research interests include computer networking, operating systems, and transparent computing. He is the Editor-in-Chief of Chinese Journal of Electronics and a fellow of the Chinese Academy of Engineering.