Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4282859, 19 pages
https://doi.org/10.1155/2022/4282859

Research Article

G/M/1-Based DDoS Attack Mitigation in 5G Ultradense Cellular Networks

Qinghang Gao,[1] **Hao Wang**,[1] **Liyong Wan**,[2] **Jianmao Xiao**,[1] **and Long Wang**[1]

[1]*School of Software, Jiangxi Normal University, Nanchang 330022, China*
[2]*Management Science and Engineering Research Center, Jiangxi Normal University, Nanchang 330022, China*

Correspondence should be addressed to Hao Wang; wanghao@jxnu.edu.cn

Received 24 January 2022; Revised 11 March 2022; Accepted 24 March 2022; Published 20 April 2022

Academic Editor: Alessandro Bazzi

With the 5G millimeter wave (mmWave) application, ultradense cellular networks are gradually becoming one of the core characteristics of 5G cellular networks. In the edge computing environment, considering load balancing among edge nodes is beneficial to slow down the process of distributed denial of service (DDoS) attack. However, most existing studies have given less consideration to congestion in the multiuser and multiedge server models. Someone who uses the M/M/1 model also seems to ignore the effect of scheduling algorithms on the Markov property of the task arrival process. In this manuscript, based on ensuring the quality of experience (QoE) for users, the G/M/1 model is introduced to the task scheduling of edge servers for the first time to improve load balancing between edge servers. For the multi armed bandit (MAB) algorithm framework, specific metrics are established to quantify the degree of its equilibrium. The number of users assigned to the edge nodes and each edge node's processing of specific tasks is taken into account. We experimentally evaluated its performance against two baseline approaches and three state-of-the-art approaches on a real-world dataset. And the experimental results validate the effectiveness of this method.

1. Introduction

As is known to all, user equipment (UE) has low computing capacity. It may not efficiently solve task requests initiated by users, while cloud services have problems such as long transmission delays. The presence of mobile edge computing (MEC) brings mobile computing, network storage, and control issues down from the cloud to the network edge, driving the execution of compute-intensive, latency-critical applications on mobile devices, effectively reducing latency and energy consumption [1–3].

There are still deficiencies in interoperability, heterogeneous architecture, data privacy, and load balancing in heterogeneous edge computing systems, which can be considered to be compensated for by requirements such as federated deployment and resource management [4]. Edge servers have limited memory, central processing unit (CPU), storage, and other resources. They generally deploy at base stations close to user terminals, and users are guaranteed low latency and stable connectivity by using edge servers [5]. The emergence of the user plane function (UPF) separates the control plane from the user plane, making MEC even more critical in 5G technology. The emergence of the 5G millimeter wave has significantly expanded the transmission bandwidth and reduced the transmission delay of mobile communications, but there are also challenges such as easy loss. Increasing the density of base stations (BSs) helps to minimize losses, and thus, ultradense cellular networks are gradually becoming one of the core features of 5G cellular networks [6]. The deployment of large-area and high-density BSs will bring new network security issues. Due to the limited signal transmission range and edge server resources, a typical IoT-based distributed denial of service (DDoS) attack can disable most nodes in a particular

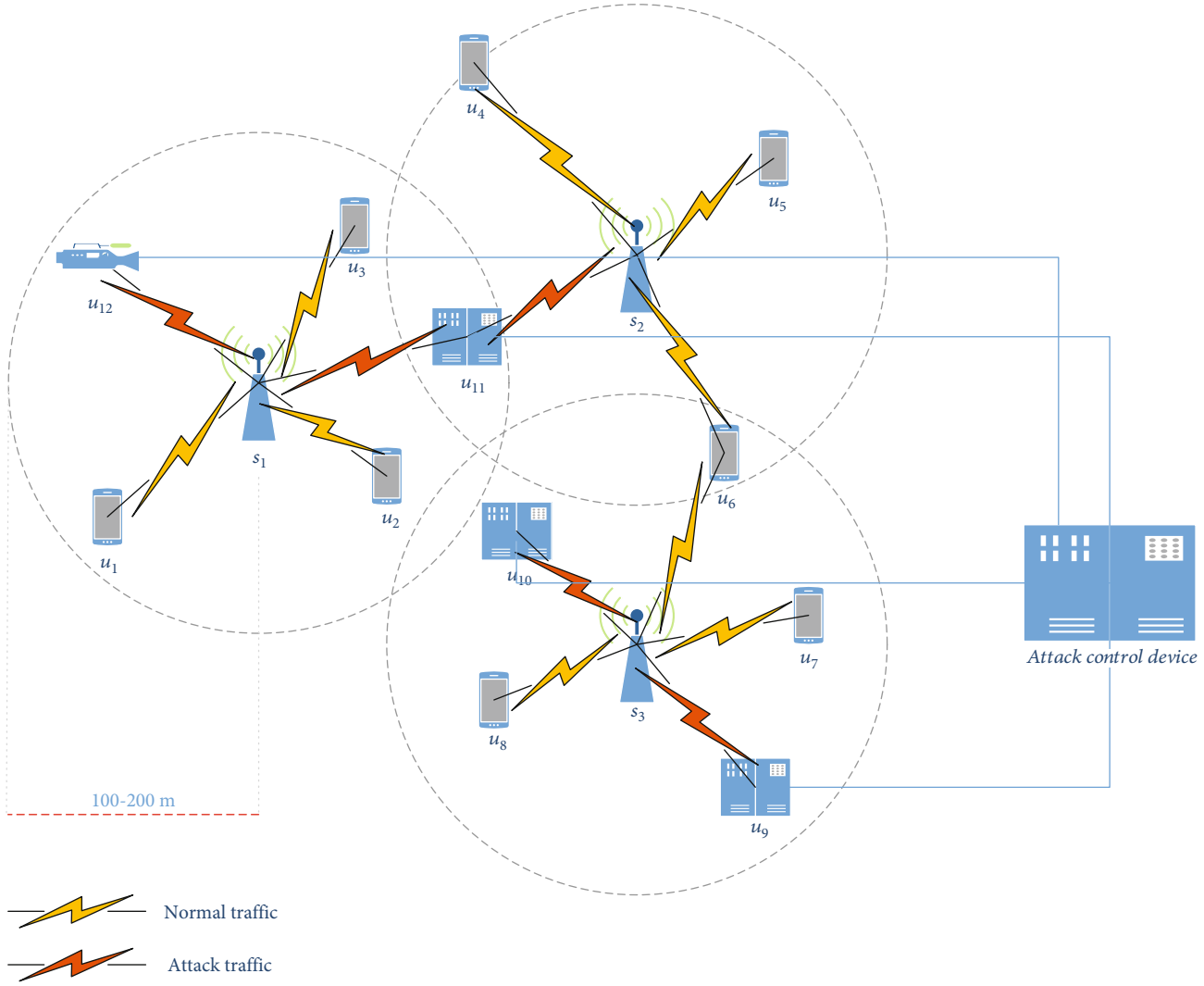


Figure 1: Ultradense cellular network topology diagram.

area by continuously trying to occupy the resources of edge nodes [7], thus paralyzing the Internet of Things (IoT) devices in its service interval within a specific period (smart monitors, infrared sensors, etc.) [8], causing severe social impacts and security problems.

DDoS attack is a resource competition problem between attackers and defenders [9], and this competition will be more prominent in resource-limited edge service environments [8]. Based on the careful consideration of system characteristics such as proximity constraint, capacity constraint, and delay constraint among edge servers [10], balancing the workload among edge servers can slow down the DDoS attack process [8], thus leaving enough reaction time for the system and reducing the possibility of the system being breached. We consider the load balancing problem of edge user allocation (EUA) based on users' quality of experience (QoE) and establish specific metrics to quantify the degree of balance. To more precisely quantify the QoE, we introduce the multi-armed bandit (MAB) algorithm framework and add nonstationary factors to the learning mechanism to better adapt to the actual complex and variable task assignment process.

For the time being, there is relatively little research on the load balancing problem and mainly reflected in the relatively balanced number of choices of edge servers [11–13]. In reality, in addition to the task volume of each mobile device which may be different, there may also be performance differences among MEC servers. Simply considering the relative uniformity of task allocation among servers may cause the performance of high-performance servers to be wasted and aggravate the waiting time and performance wear and tear of less performing servers. In studies involving user task waiting time (stay time), they are mainly divided into two forms: computational time accrual for queueing tasks [14] [15] [13] and the use of the M/M/1 queuing model [16] [17]. For the latter, researchers have ignored the effect of the scheduling algorithm on the Markov property of the M/M/1 queuing model, i.e., subjective task scheduling that undermines the principle of no posteriority of the task arrival process.

Require: C_{max}, $B \longrightarrow \infty$, $\delta = 1$, $\eta = 0$, σ
1: **for** $s_i \in S$ **do**
2: Generate uniformly distributed random variables λ_i, μ_i, ζ_i from [0,1)
3: **if** $\mu_i \leq \lambda_i$ **then**
4: $\mu_i \leftrightarrow \lambda_i$
5: **end if**
6: $C_{i,n} = \delta((a_j/B) + (l_{j,i}/v) + (1/\mu_i(1-\zeta_i))) + \eta \cdot \kappa \cdot \phi_{i,n}$
7: **end for**
8: Generate normally distributed random variables α_i, β_i with (1, 0.5)
9: **while** $t \leq T$ **do**
10: **for** $s_i \in S$ **do**
11: $\theta_i \longleftarrow Beta(\alpha_i, \beta_i)$
12: **end for**
13: $s_j(t) \longleftarrow argmax\, \theta_i \forall s_i \in S$
14: $n_j = n_j + 1$
15: Generate uniformly distributed random variable U from [0,1)
16: $c_{j,t} = -\ln(1-U)/\mu_j(1-\zeta_j)$
17: $r_j{}' = 1 - p = 1 - (c_{j,t}/c_{\max})(p \leq 1)$
18: $r_j \longleftarrow r_j + \sigma(r_j{}' - r_j)$
19: $\mu_j \longleftarrow \mu_j + (r_j{}'/n_j)$
20: $(\alpha_k, \beta_k) = \begin{cases} (\alpha_k, \beta_k) \text{if } s_j(t)=k \\ (\alpha_k + r_j, \beta_k + 1 - r_j) \text{if } s_j(t) = k \end{cases}$
21: $t = t + 1$
22: **end while**

Algorithm 1: Thompson sampling nonstationary (TSNS).

Queuing systems generally consist of customers, service desks, and queuing rules [18]. Under conditions independent of other factors, a customer's arrival satisfies Poisson distribution, the service time satisfies negative exponential distribution, and a single service desk processes the task of the customer, those situations that can be represented by the M/M/1 model [18–20]. A customer's arrival is usually independent of others, while the service desk is responsible for solving the task requests of arriving customers. The processing time of specific tasks is influenced by stochastic factors such as the nature of the customer's task and the service desk. When we use the algorithm to schedule the user assignment process in the edge environment, customer arrivals will no longer follow the Poisson distribution, and continuing to use the M/M/1 model at this point seems to deviate from reality. Regarding the G/M/1 model, the process of customer's arrival is not restricted, which is "general arrival," and the service process still follows a negative exponential distribution, which considers the randomness of customer tasks and service desks [18–20].

In the actual edge user assignment process, we reduce the impact of the scheduling algorithm on the Markov property of the edge server's task arrival process by applying the G/M/1 queuing theory model. And a nonstationary factor is added to the MAB algorithm framework to consider the impact of the edge server's task processing capacity fluctuation on the computation delay. In an attempt to improve load balancing in the edge user assignment scenario to mitigate the DDoS attack process, we further consider factors such as the number of tasks offloaded by edge users, possible performance differences among edge servers, and more in-depth consideration of the specific task processing of each edge node. To enhance the experiment's credibility, we used a real dataset from the Central Business District (CBD) of Melbourne [21] and compared it extensively with existing studies, and the experimental results verified the effectiveness of the algorithm. The main contributions are as follows:

(i) We attempt to improve the load balancing for edge user allocation in edge computing to slow down the DDoS attack process

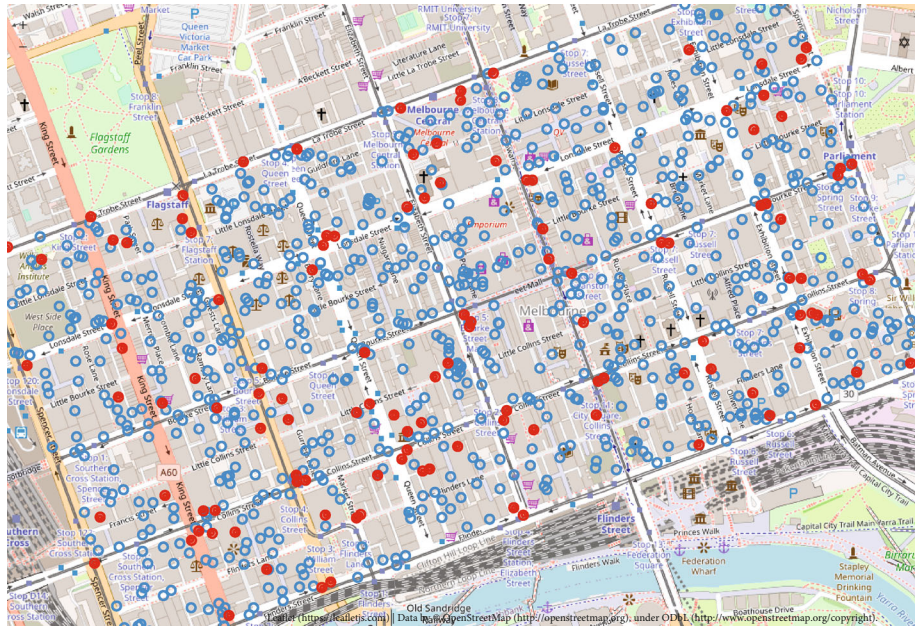
(ii) This is the first attempt to study the EUA problem through the MAB algorithm framework in 5G ultra-dense cellular networks, considering the processing of specific tasks in each edge node. The number of users in edge nodes is no longer considered solely

(iii) This is the first attempt to introduce the G/M/1 queuing theory model to the MEC system, considering the impact of scheduling algorithms on the Markov property of the actual task arrival process. And the performance is experimentally evaluated on a widely used real-world dataset

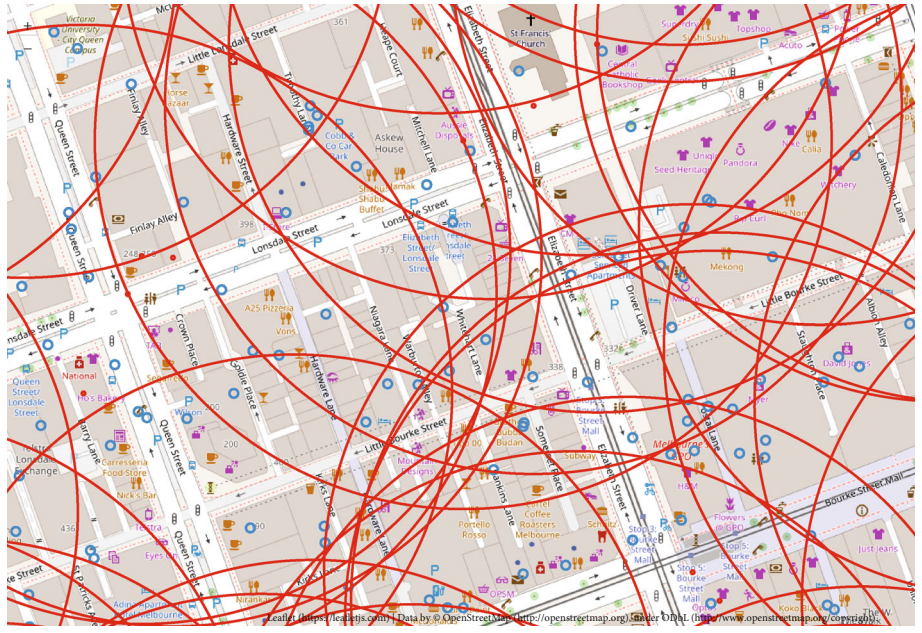
The rest of the paper is organized as follows. Section 2 presents related work. Section 3 offers the system model. Section 4 proposes the Thompson sampling nonstationary (TSNS) algorithm. Section 5 designs experiments and evaluates the algorithm's performance, and Section 6 concludes the paper.

2. Related Work

Currently, relatively little research has been done on DDoS attack in edge computing, mainly in edge collaboration, attack identification, and defense [8, 22–27]. The literature [8] studied the DDoS attack mitigation problem in edge computing, proved its NP-hardness, and proposed a game-theoretic approach to solving the problem. The literature [22] considered mitigating the DDoS attack process by balancing the incoming control plane's total traffic and inducing the attack initiator to stop the attack. The literature [23] designed an adaptive traffic scheduling algorithm to enhance collaboration among edge nodes and thus reduce DDoS attack. The literature [24] developed an intrusion detection and defense method for edge environments by learning the original data distribution through Deep Convolution Neural Network (DCNN) and building defense through Q-network algorithm. In a software-defined network (SDN), [25] established initial detection of intrusion based on entropy and further accurate detection by integrated learning, reducing communication overhead and attack detection latency. From the perspective of smart cities, [26] on the fractional-level fusion of multimodal biometrics effectively improves recognition accuracy and [27]



(a) Point distribution



(b) Relationships

Figure 2: Dataset for Melbourne's Central Business District.

proposed a data encryption technique applicable to the IoT, etc.

The low latency of edge computing is fundamental for users to execute resource-intensive and latency-sensitive applications on edge devices. It is a crucial factor affecting the QoE of user experience [4]. In the study of resource allocation for computational offloading, with the goal of latency optimization, [28–31] schedule computational tasks through a Markov decision process, [14, 32, 33] consider game theory to obtain the best strategy for task offloading, and [11, 12, 34–39] consider algorithms such as reinforcement learning to solve problems related to resource allocation. The literatures [13, 15–17] introduced the MAB algorithm framework to learn online to adjust task allocation in real time. Among them, only the literatures [11–14, 34] consider the load balancing problem of edge servers from the perspective of resource allocation.

The literature [14] proposed a decentralized learning algorithm from a game-theoretic perspective, considering a relatively uniform number of users allocated across edge servers. However, the experimental design with the same upper bound of acceptable cost for users may deviate from reality. From the perspective of on-edge computing, [11] transformed the offloading and load balancing problem into

Table 1: MEC servers.

Parameter	s_1	s_2	s_3	s_4	s_5
μ	0.4250	0.8865	0.6233	0.9485	0.9973
ζ	0.0735	0.7655	0.2477	0.8558	0.9471
Parameter	s_6	s_7	s_8	s_9	s_{10}
μ	0.7121	0.9000	0.8639	0.6185	0.8444
ζ	0.9921	0.0281	0.9425	0.4937	0.3638
Parameter	.....	s_{122}	s_{123}	s_{124}	s_{125}
μ	.....	0.7192	0.9712	0.1817	0.3955
ζ	.....	0.8388	0.2631	0.6614	0.4781

a mixed-integer nonlinear programming problem and changed the problem into two subproblems for optimization. The literature does not seem to consider the effect of the waiting factor, and the possible case of multiple tasks appearing at the same node is not further explored. It is only described from a collision perspective. The literature [34] introduced fiber-wireless (Fi-Wi) technology to enhance the signals of vehicular edge computing networks (VECNs), which in turn use software-defined networking (SDN) to achieve load balancing. The literature considers the possibility of task assignment locally, at the edge nodes or in the cloud. Still, the impact of the coverage of the signaling edge nodes may be neglected in the selection process of the offload servers, and task processing at the edge nodes seems to lack consideration of congestion factors. The literature [39] utilized multipath TCP to increase application throughput and used reinforcement learning-empowered multipath manager to address the buffer congestion problem further. In the literature [12], on the premise of determining the set of optional edge service nodes for each mobile device users (MDUs), the situation of the user devices to be assigned to each edge node and their computational capabilities were considered comprehensively, and new devices were assigned to the edge server with less computational pressure accordingly. The algorithm design process seems to ignore the influence of congestion factors within the edge nodes, and the optimal edge server may deviate from the actual scenario only in terms of transmission and computation delay; i.e., there may be a large waiting delay after the task arrives at the redistributed edge node. In addition, there may be a significant task assignment delay. Uncertainty decision-making is an essential challenge in machine learning, and the MAB algorithm is a common framework for solving this problem, where each MEC server is considered an arm [40]. The literature [13] proposed a utility table-based MAB algorithm with online learning to adjust the workload allocation in real time and update the feedback signal after task allocation through the utility table to determine the optimal solution. The literature mainly considers load balancing from cloud-edge collaboration and gives less consideration to task allocation among edge servers.

As we know, the DDoS attack problem is currently a hot topic of research in network security, and relatively little research has been conducted from the perspective of edge computing. In edge computing, considering the load balancing of edge user allocation, we make the first attempt to study the EUA problem in 5G ultradense cellular networks through the MAB algorithm framework, focusing on the processing of specific tasks at each edge node. We made the first attempt to introduce the G/M/1 queuing theory model into the MEC system, considering the impact of the scheduling algorithm on the Markov property of the actual task arrival process.

3. System Model

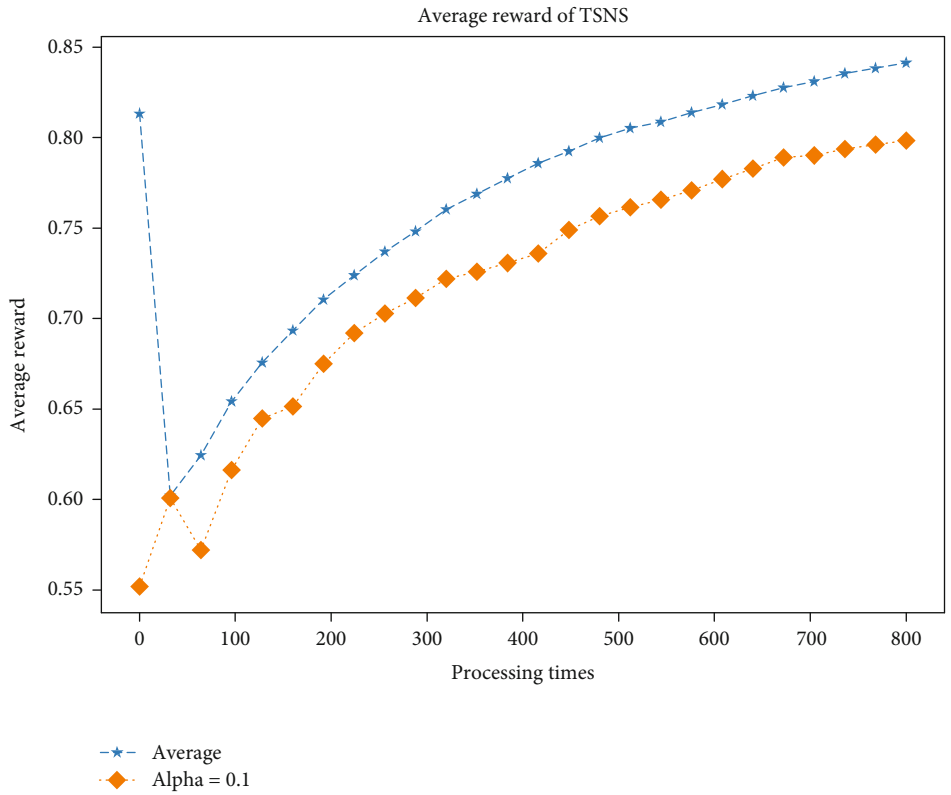
In the ultradense cellular network scenario, in the edge user allocation process, as in Figure 1, we use $s_i \in S$ to denote the set of MEC servers and $u_i \in U$ to represent the set of user devices. In edge computing, in addition to service requests from regular users, DDoS attack can launch frequent task requests to the edge server by controlling multiple IoT devices in the service range. Considering the influence of the scheduling algorithm on the Markov property of the resource allocation process, we assume that the task arrival process follows a general distribution and the service time follows a negative exponential distribution. Since each MEC server has different task arrival and service capacity and there is no restriction on queue length and task origin, the task offloading process can be represented by the G/M/1 queueing theory model. We denote by ζ_i the task arrival impact factor per unit time of server s_i, which can be obtained by solving the scheduling process and by μ_i the average service rate of server s_i. Every time a task assignment is made, the average service rate of the selected server is updated by a nonstationary method.

We consider $C_{i,n}$ to denote the cost of processing task n for server i and $D_{i,n}$ and $E_{i,n}$ to denote the corresponding service latency and energy consumption. Therefore, $C_{i,n}$ is computed as follows:

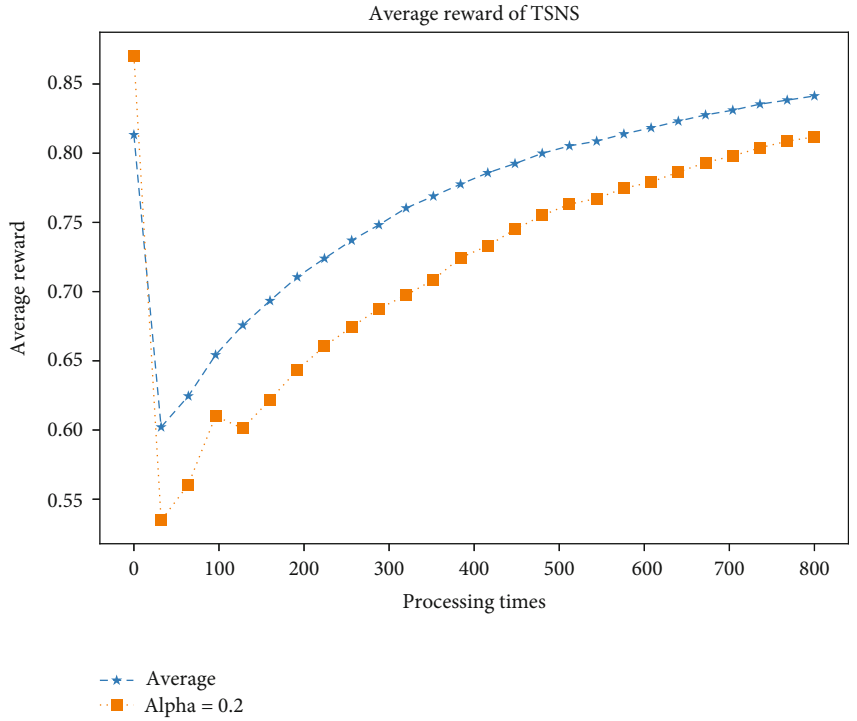
$$C_{i,n} = \delta \cdot D_{i,n} + \eta \cdot E_{i,n}. \tag{1}$$

In the formula, δ and η, respectively, represent the weight of delay and energy consumption in the cost, $\delta + \eta = 1$.

We understand that in 5G ultradense cellular network architecture, the physical distance between microcell BSs is typically between 100 and 200 m [6]. The delay can be further subdivided into transmission delay, propagation delay, waiting delay, and computation delay. The sum of the waiting and computation delays is the delay of the task staying in the system. The signal strength decreases from the central node in all directions [5]. We may assume that the effective signal coverage of each edge node is 200 m (edge servers beyond the signal range can be selected, but the selection cost is relatively high [5]), and on this basis, we consider the limited nature of each user when selecting an edge server. And since the physical distance $l_{i,j}$ of the computing task from the user end u_i to the edge node s_j generally does not exceed 200 m, its actual propagation delay will be at the microsecond level. We usually use the ratio of task volume a_j

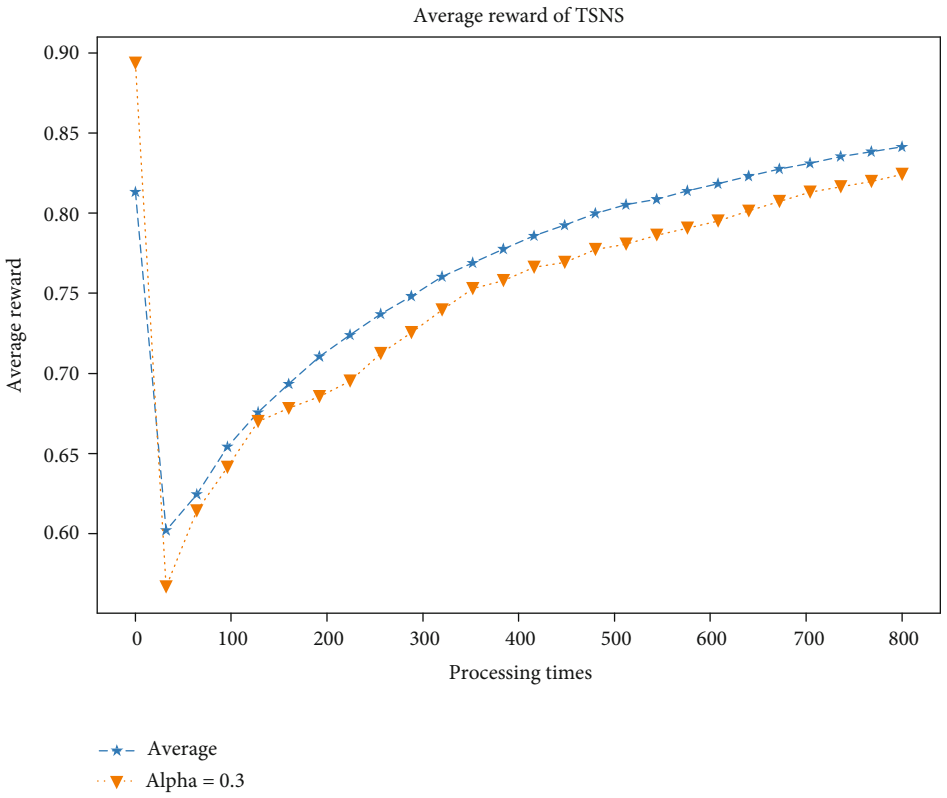


(a) $\sigma = 0.1$

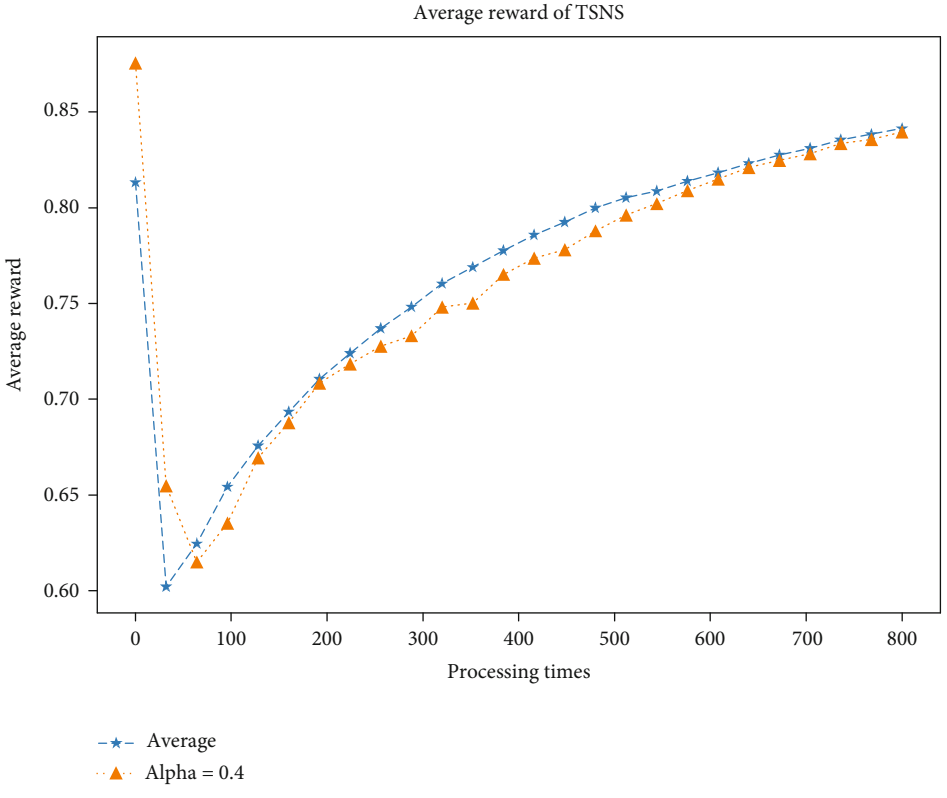


(b) $\sigma = 0.2$

Figure 3: Continued.

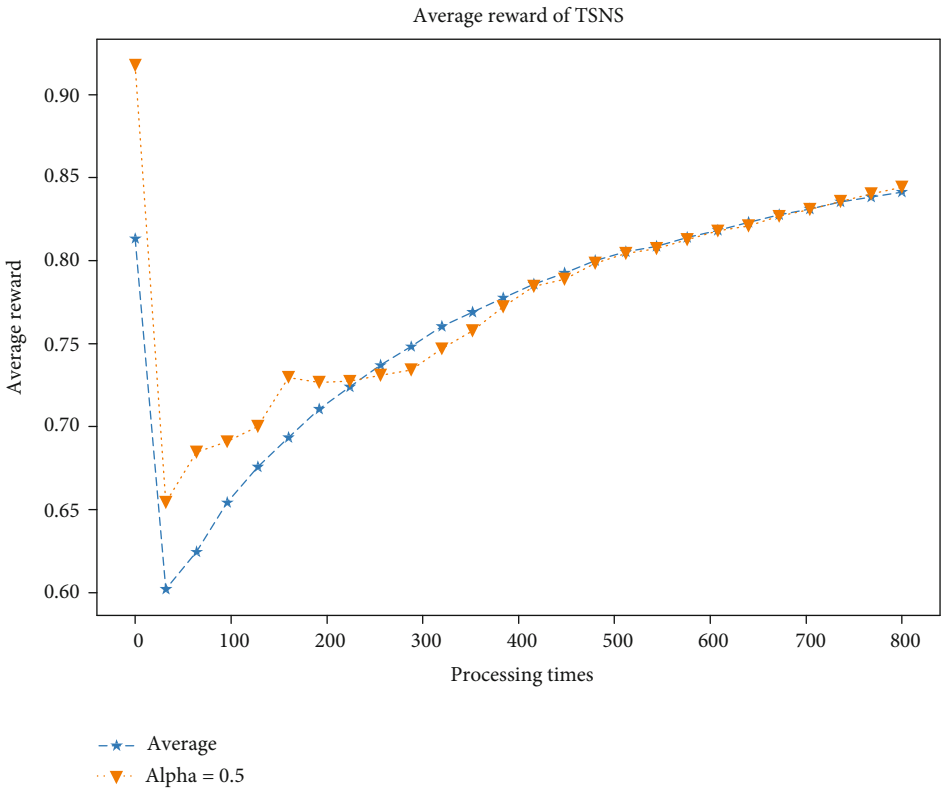


(c) $\sigma = 0.3$

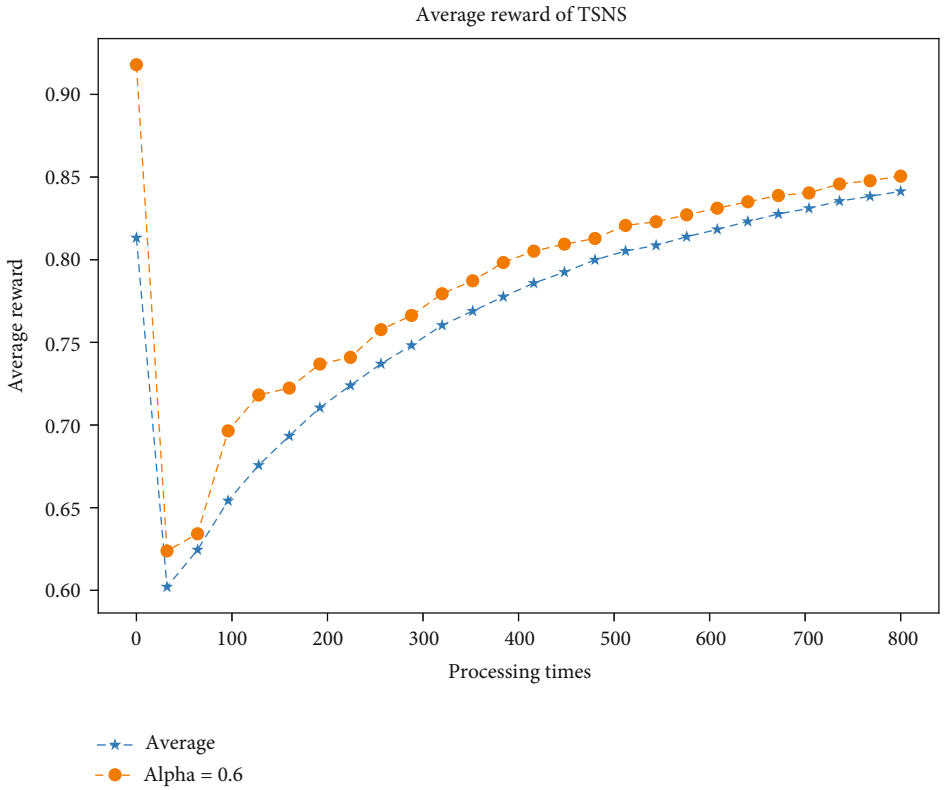


(d) $\sigma = 0.4$

Figure 3: Continued.

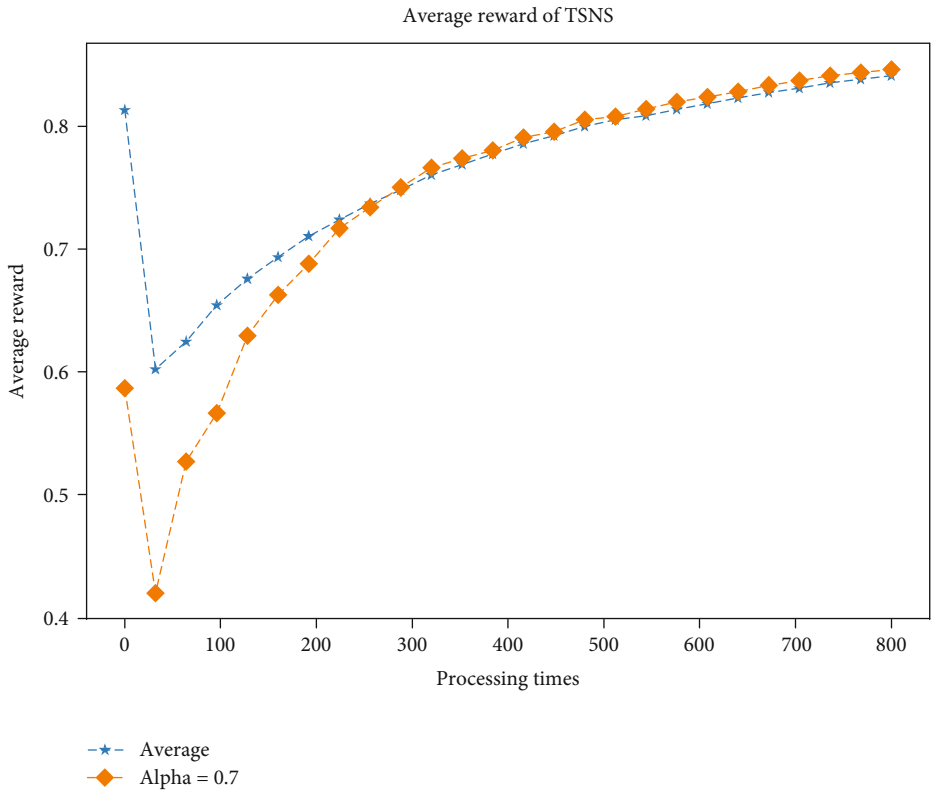


(e) $\sigma = 0.5$

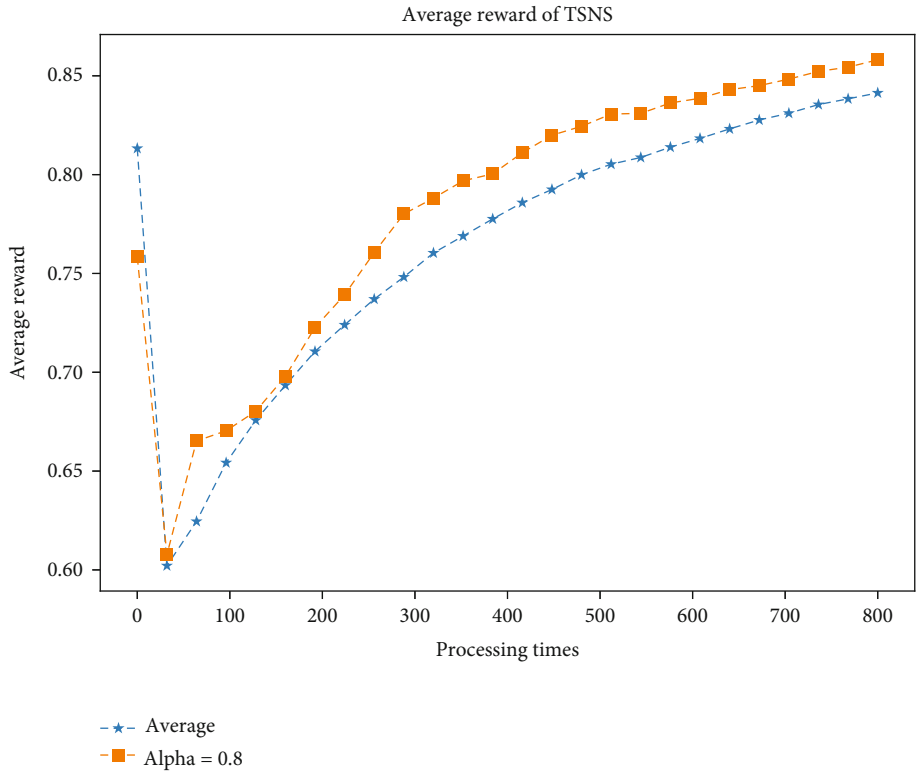


(f) $\sigma = 0.6$

Figure 3: Continued.

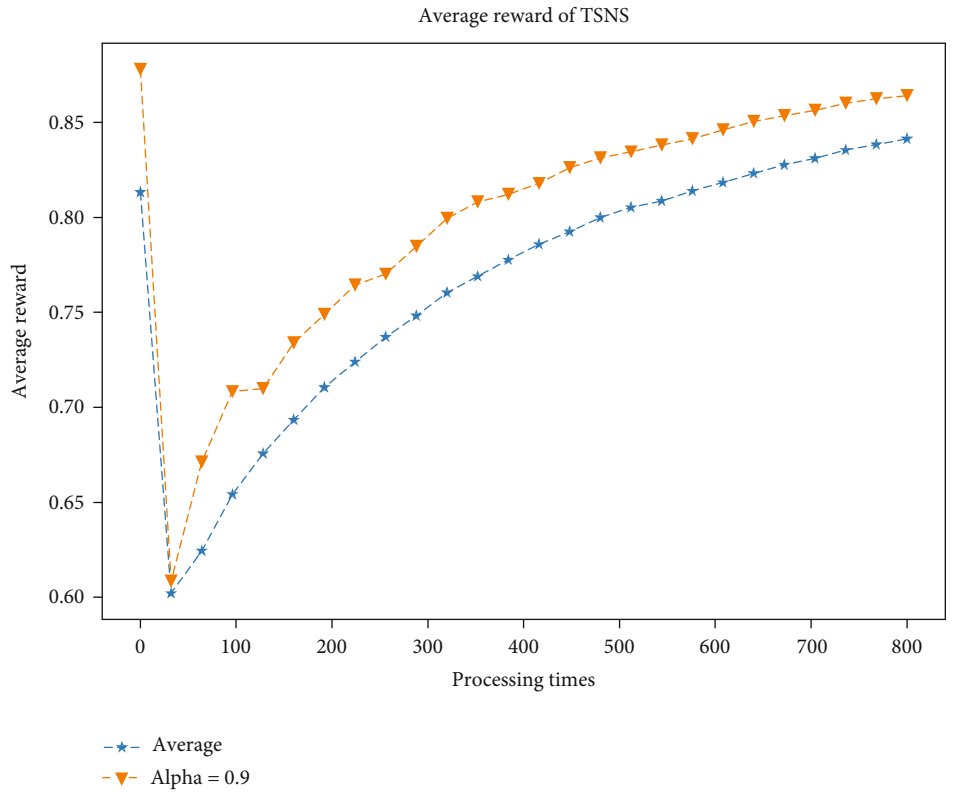


(g) $\sigma = 0.7$



(h) $\sigma = 0.8$

Figure 3: Continued.



(i) $\sigma = 0.9$

Figure 3: Different trends of the average reward in different alpha.

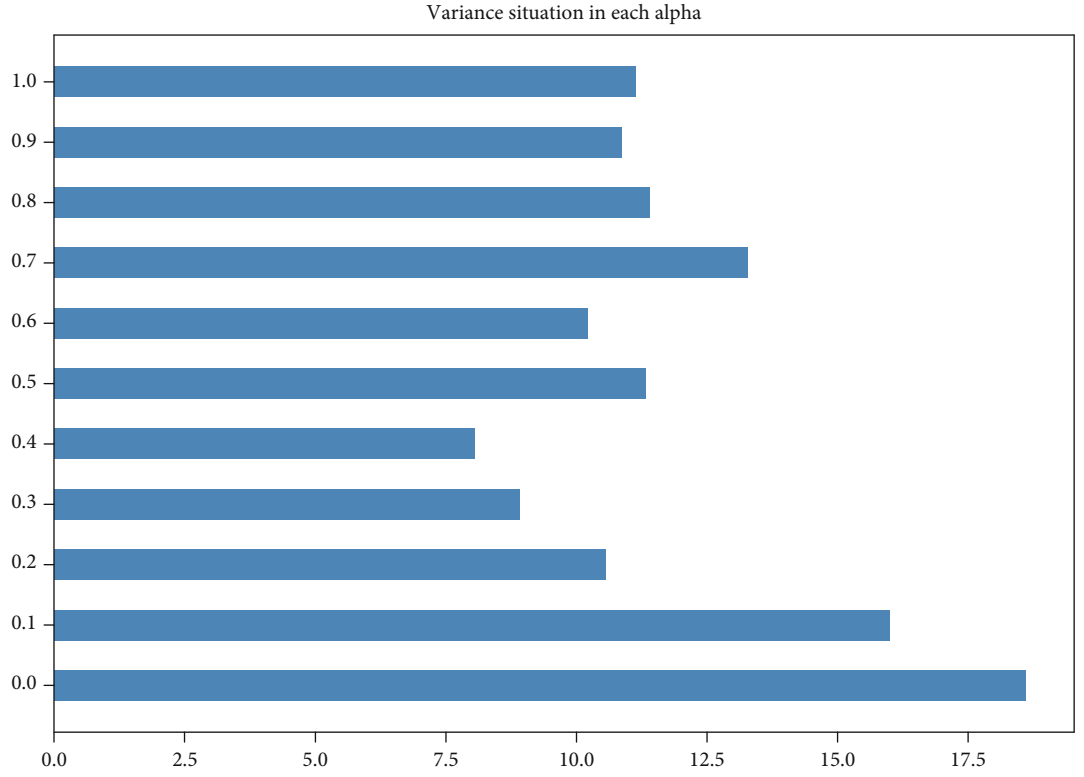


Figure 4: The variance in the TSNS algorithm.

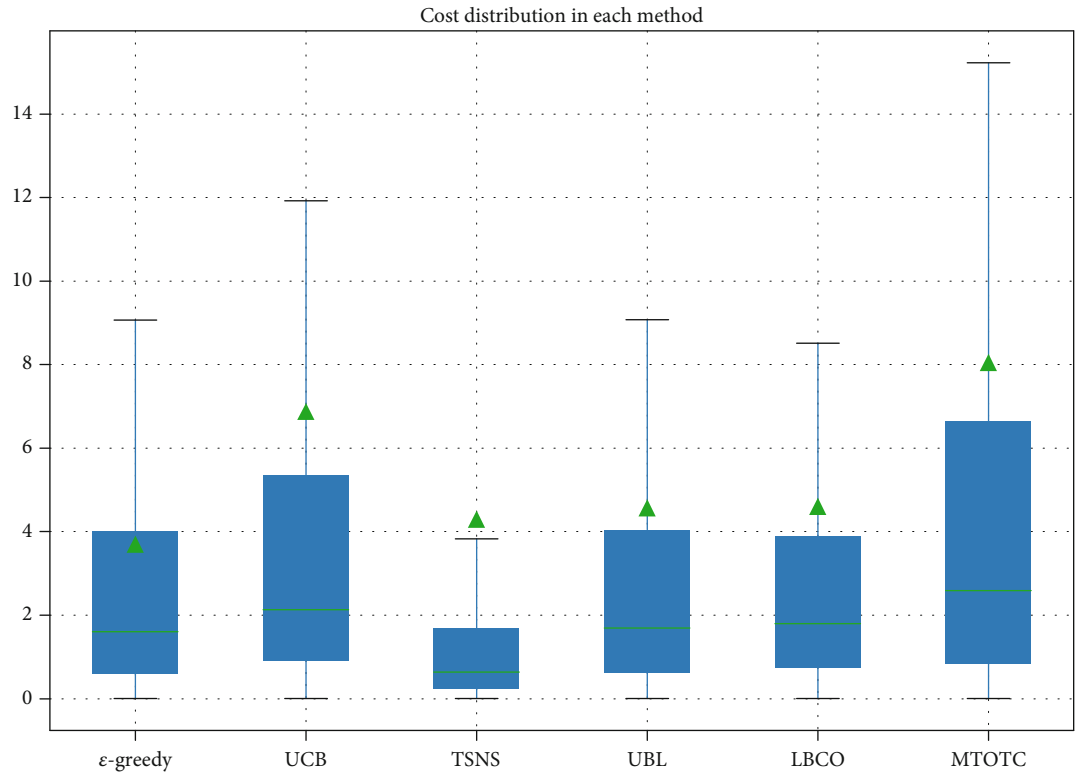


Figure 5: Boxplot for the cost distribution.

to channel bandwidth B to express the transmission delay. In the ultradense cellular network structure, the case of transition node forwarding tasks is basically nonexistent; i.e., the transmission delay of tasks will also be close to a subtle level. According to the queuing theory [18–20], the stay time T obeys the distribution $P\{T \le t\} = 1 - e^{-\mu(1-\zeta)t}$, whose average stay time is $1/(\mu(1-\zeta))$, and the actual stay time of each task can be randomized by the distribution function. We use κ to represent the energy consumption influencing factor that comprehensively considers power, signal-to-noise ratio, and other factors. Let $\phi_{i,n}$ denote the task size of task n in server s_i, and the energy consumption is $E_{i,n} = \kappa \cdot \phi_{i,n}$ [41, 42]. Further, we can get the following formula:

$$C_{i,n} = \delta\left(\frac{a_j}{B} + \frac{l_{j,i}}{v} + \frac{1}{\mu_i(1-\zeta_i)}\right) + \eta \cdot \kappa \cdot \phi_{i,n}, \tag{2}$$

where a_j denotes the number of tasks to be processed by user device u_j, which in general is equal to $\phi_{i,n}$. B is the channel bandwidth, and a_j/B is the transmission delay. $l_{j,i}$ denotes the physical distance between user device u_j and edge node s_i. v indicates the propagation speed of the task in the channel, which is generally equal to or slightly less than the speed of light, and $l_{j,i}/v$ is propagation delay.

To measure the QoE more specifically, we introduced the MAB algorithm framework. An upper bound on the cost is chosen as $C_{\max}$, and we assume that the cost as a percentage of the given threshold is p. The reward after each selection can be calculated as follows:

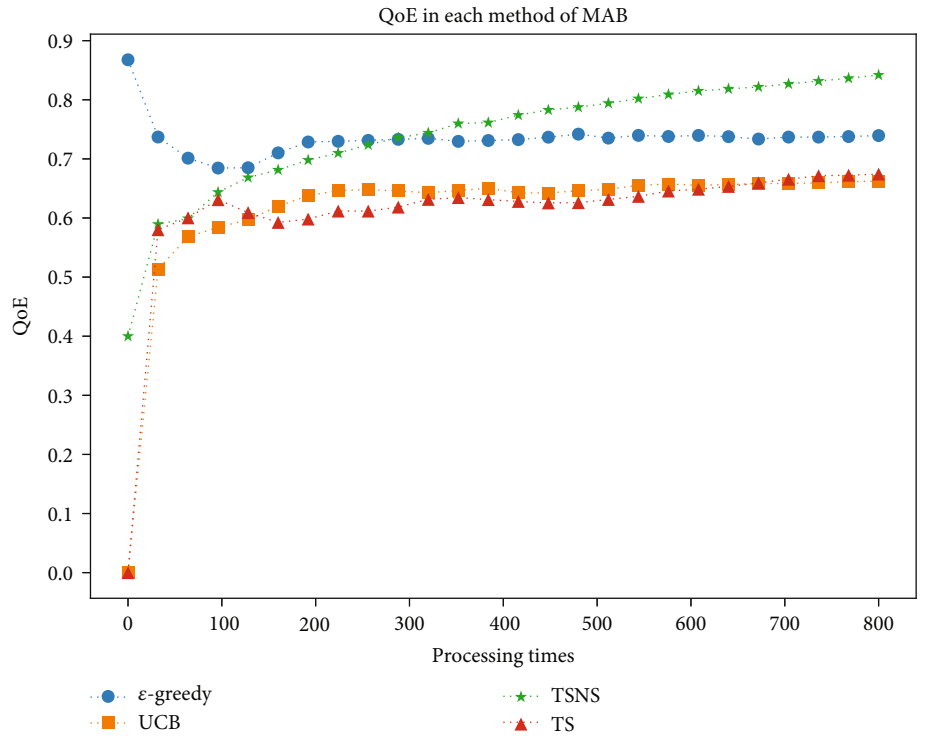
$$R_{i,n} = (1-p) \cdot 1_{(C \le C_{\max})}, \tag{3}$$

where 1 is the indicator function.

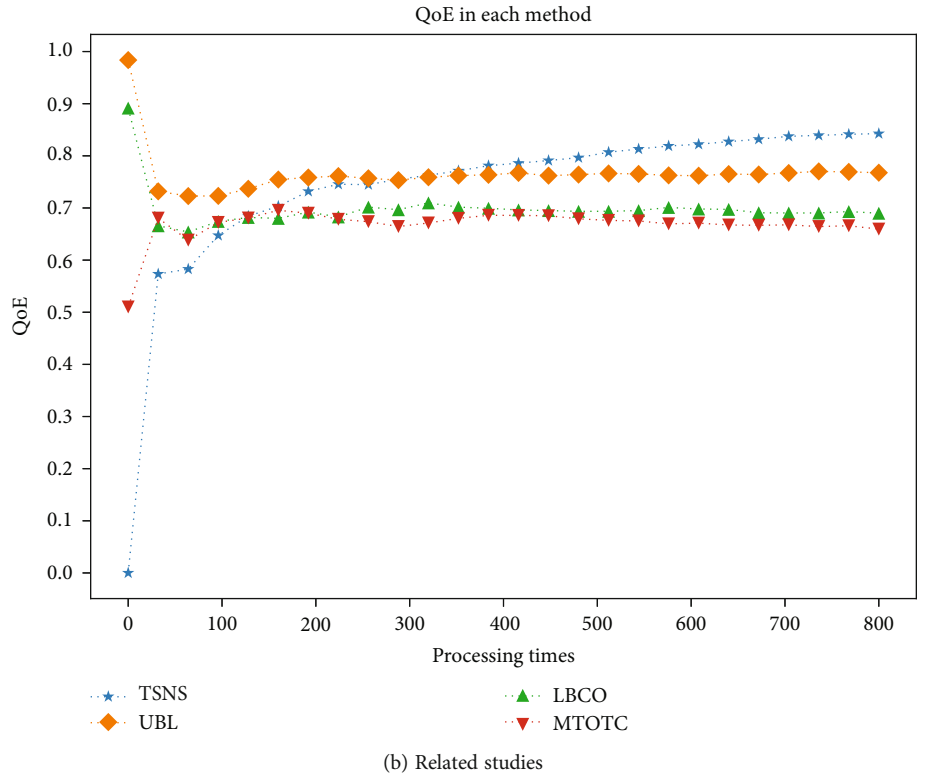
4. Algorithm Design

The MAB model is a simple but compelling algorithmic framework that can make decisions over time in uncertain situations [43]. It simulates an agent, learning new knowledge to optimize selection decisions.

We know that considering load balancing in the edge environment is beneficial to slow down the DDoS attack process [8, 22]. We use the MAB algorithm framework to balance the limited task processing latency and cost and offload the tasks to each MEC server as evenly as possible. Each MEC server can be considered an arm of varying nature, and each selection of the arm can be rewarded and cost accordingly. This property is unknown to the task assignor, so we may call it an implicit property. As the number of selections increases, the resource allocation of edge servers will become more rational, and the number of tasks processed per unit time will improve. In addition, considering the complexity and variability of the actual task arrival and processing, the server's performance may also change with the increasing number of selections, and we introduce a nonstationary factor.



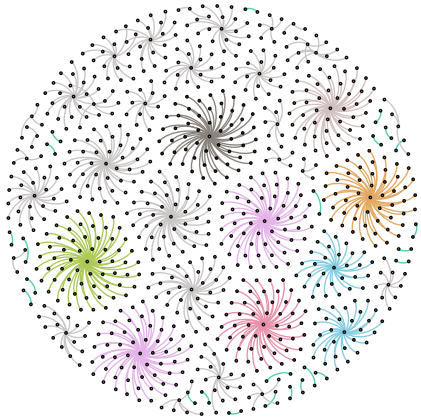
(a) In MAB



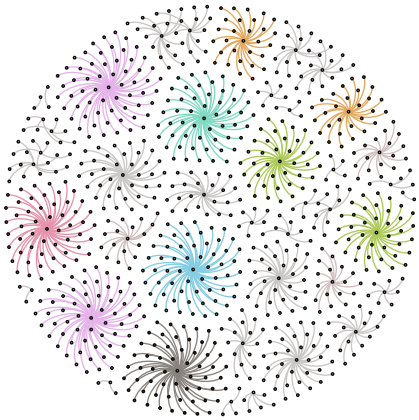
(b) Related studies

Figure 6: QoE values during the selection process.

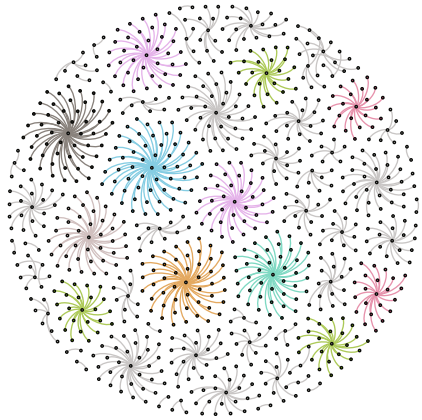
To reduce useless exploration and increase the exploration of the arm with larger pairwise differences, we consider applying the improved Thompson sampling to the MAB algorithm. In the Thompson sampling algorithm, the payoff value of each action follows a beta distribution, with α and β as prior probability parameters. All the arms will generate a random number as payoff value through beta distribution according to their prior probability parameters whenever a selection is made. The system will select the arm with the largest payoff value. The probability distribution law of



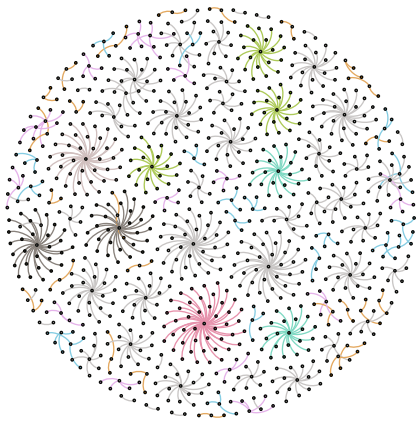
(a) UBL



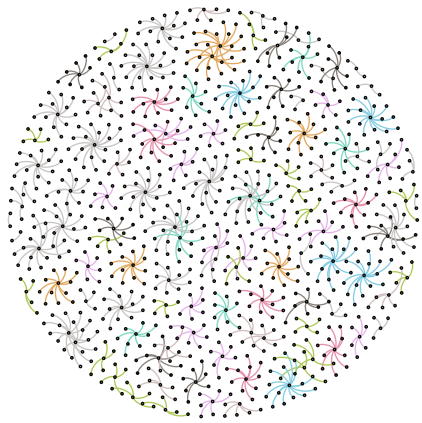
(b) LBCO



(c) Improved ε-greedy

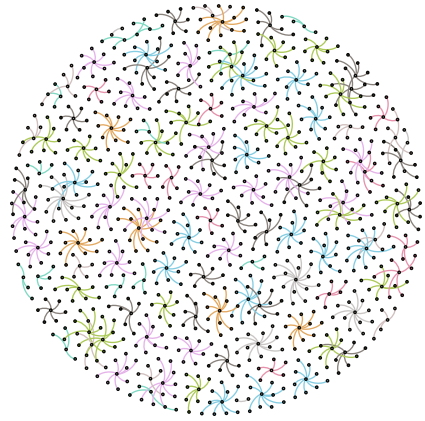


(d) TSNS



(e) MTOTC

Figure 7: Continued.



(f) UCB

Figure 7: Relationship between edge nodes and user devices.

Bernoulli distribution and the probability density of beta distribution are as follows:

$$p(x) = \theta^x (1-\theta)^{1-x}, \quad x = 0, 1, \tag{4}$$

$$p(\theta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}, \quad \theta \in [0, 1], \tag{5}$$

where the two refer to the distribution of returns and the distribution of the parameter θ of the return distribution. $\Gamma(z)$ satisfies the formula

$$\Gamma(z) = \int_0^{\infty} x^{z-1} e^{-x} dx, \quad R(z) > 0. \tag{6}$$

We assume that there are S edge servers, and T tasks are processed in a certain period of time. Each selection will update the distribution. When action k is selected, the return is subject to a Bernoulli distribution with parameter θ_k.The probability of returning 1 is θ_k, and returning 0 is $1-\theta_k, \theta = (\theta_1, \theta_2, \theta_s)$. In round t, select the action $a_t \in \{1, 2, s\}$ will receive a return $r_t \in (0, 1)$. Assuming that θ_k are independent of each other, the prior distribution obeys beta(α_k, β_k), and the posterior distribution obeys beta$(\alpha_k + r_t, \beta_k + 1 - r_t)$.

$$p(\theta_k) \propto \theta_k^{\alpha_k - 1}(1-\theta_k)^{\beta_k - 1}, \tag{7}$$

$$p(\theta_k | r_t) \propto \theta_k^{r_t}(1-\theta_k)^{1-r_t}\theta_k^{\alpha_k-1}(1-\theta_k)^{\beta_k-1} = \theta_k^{\alpha_k+r_t-1}(1-\theta_k)^{\beta_k+1-r_t-1} \tag{8}$$

For each selection made, the parameters of the posterior distribution of the selected arm will be calculated based on its return values. The posterior distribution of the last round can be used as the prior distribution of the next round, and the parameter update rule of the posterior distribution beta is [44]

$$(\alpha_k, \beta_k) = \begin{cases} (\alpha_k, \beta_k) & \text{if } a_t \neq k, \\ (\alpha_k + r_t, \beta_k + 1 - r_t) & \text{if } a_t = k. \end{cases} \tag{9}$$

We use the reward $R_{i,n}$ of the edge nodes after performing the task processing as the QoE measure for the corresponding users. As we analyzed above, the propagation delay and transmission delay under delay segmentation is at the microsecond level, which is negligible compared to the task's computation delay and queuing delay. In contrast, the task processing energy consumption is a weak user experience. To simplify the model, we set $\eta = 0$ and $\delta = 1$, and the channel bandwidth is infinite concerning the task volume and mainly considers the average stay time of the task in the system. After each selection, we add a nonstationary utility learning mechanism [45].

$$Q_{i,n} = Q_{i,n-1} + \gamma(R_{i,n} - Q_{i,n-1}), \tag{10}$$

where γ represents the learning rate in the selection process; i.e., the greater the γ, the greater the importance of the actual reward, and the greater the degree of learning in calculating the utility reward. The updated utility reward is used as the reward value. In particular, after each selection, the average service rate of the selected service desk is optimized to simulate the effect of random factors in the user assignment process.

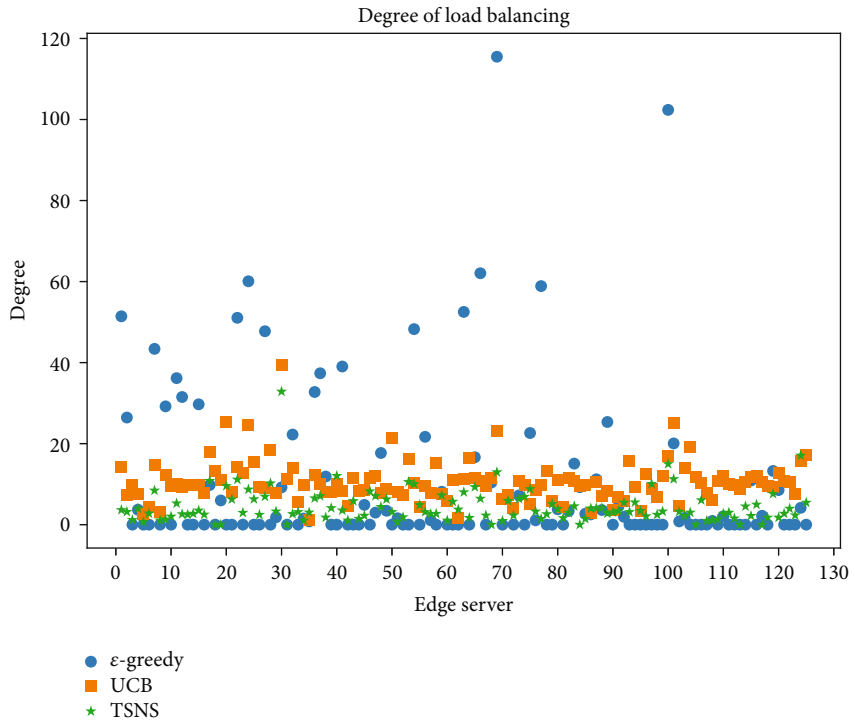
$$\mu_{i,n+1} = \mu_{i,n} + \frac{R_{i,n}}{n}. \tag{11}$$

The arm with the largest parameter θ is considered during each selection, and the calculated reward of the actual choice is used to update the posterior distribution parameters beta. The corresponding regret value is

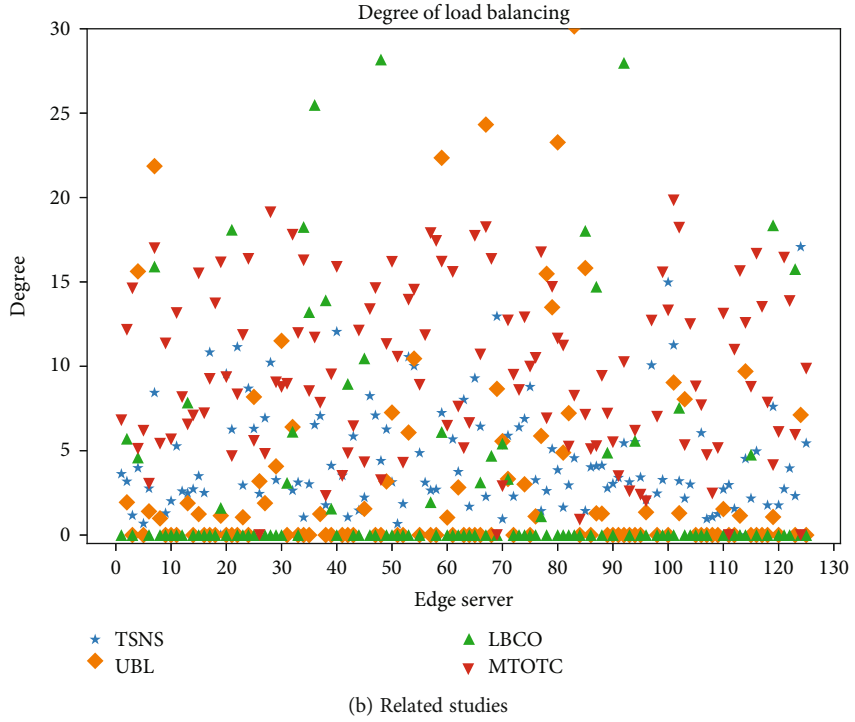
$$R(T) = \sum_{t=1}^{T} C(a_t) - \sum_{t=1}^{T} C_t^*(a_t \in S), \tag{12}$$

where a_t represents the edge server selected for time t and $C(a_t)$ is the corresponding cost of the currently selected server. C_t^* denotes the minimum value of the corresponding cost of each edge server at time t.

The specific idea of the TSNS algorithm is represented in Algorithm 1 in an ordered manner. Each user assignment is



(a) In MAB



(b) Related studies

Figure 8: Accumulated computation time in each edge node.

made that the corresponding service time and stay time are calculated according to the G/M/1 queuing theory model. Service time is an essential statistic for measuring load balancing. Stay time can be used to calculate rewards and, in turn, utility rewards.

We learn and record the specific situation after each task assignment through the MAB algorithm framework, including the actual cost and reward after each user assignment and the actual task processing latency of edge nodes, which can measure QoE and load balancing more precisely and

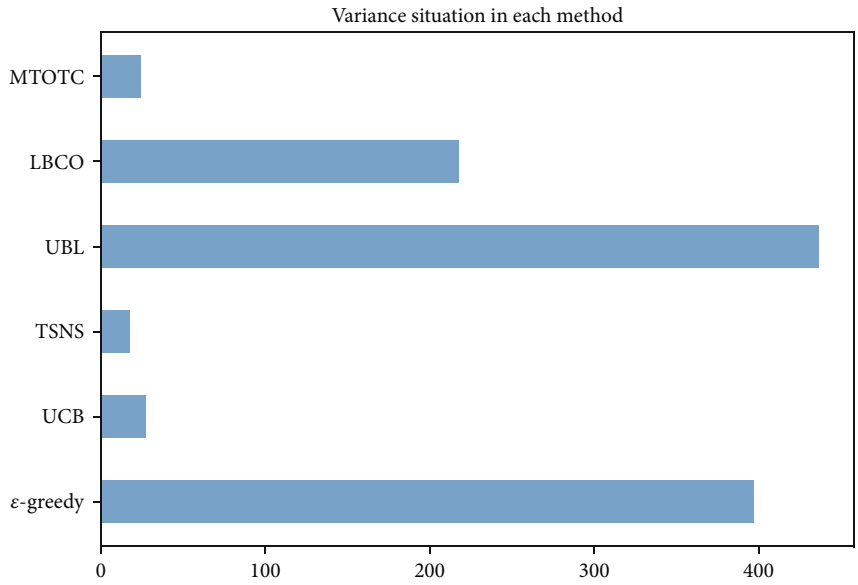


Figure 9: Load profile in each edge node.

effectively. The specific experiments are described in detail in Section 5.

5. Performance Evaluation

In this section, we conducted extensive experiments to evaluate the TSNS algorithm based on a real-world dataset from the CBD of Melbourne (e.g., Figure 2). The excerpted dataset contains 125 service base stations and 816 random users. We model the difference in task volume between users using a normal distribution with mean 10 and variance 2, in which the ratio of the random number to the mean is used as the weight of the effect of task volume on processing latency and use this as the basis for a series of experiments.

5.1. Preferences. Combining the ideas of the Monte Carlo method, we conducted an experimental design. First, we record and calculate the average stay time as the initial property of the corresponding service desk, in which the average service rate and the task arrival impact factor per unit time are solved by a uniform distribution in the interval [0,1), as in Table 1 (the experiment contains but is not limited to the parameters given in Table 1). Based on the nature of queuing theory regarding the G/M/1 model, we obtain the distribution function of the task sojourn time and consider randomizing the essential stay time of the current task at the selected server by the distribution function. For each user's specific task, we assume that it satisfies a normal distribution with a mean of 10 and a variance of 2. The ratio of the random value to the mean is used as the influence of the stay time for the specific task. That is, for each task assignment, the corresponding edge server randomizes the corresponding sojourn time for calculating the reward and, in turn, the utility reward. The calculated utility rewards are used to update the parameters of the posterior distribution beta, which in turn affects the next round of task assignment.

Regarding the initial prior distribution corresponding to each server in the TSNS algorithm, it is considered randomized out through a normal distribution with a mean of 1 and a variance of 0.5. In each task assignment process, the posterior distribution will be used as the prior distribution corresponding to the following selection, and the parameters θ_i will be randomized through the prior distribution. Then, the server with the largest parameters will be selected for the task processing.

First, we assume that the learning parameter $\sigma = 0.5$ and the cost upper bound $C_{\max} = 20$. During the selection process of the simulated edge servers, we obtained the upper quartile (Q_3), median, and lower quartile (Q_1) of the corresponding cost distribution for each method. We calculated the maximum observed value of the upper edge by $Q_3 + 1.5 * (Q_3 - Q_1)$ [46]. Subsequently, we averaged the upper edge observations for all methods and calculated an approximate cost upper bound of 10. Further, we compared the average reward profile under different learning parameters and obtained the average profile after removing the anomalous profile $\sigma = 0$. In multiple experiments, the larger the parameter σ, the better the distribution of the average reward might be, and $\sigma = 0.4$ basically fluctuates up and down around the average curve, as shown in Figure 3. We simulated the user assignment process under different parameters and obtained the variance comparison among the edge nodes, as shown in Figure 4. To balance the load situation of the server, we might as well set it as the experimental parameter. Comparison of cost distribution among methods for the upper cost bound $C_{\max} = 10$ and learning parameter $\sigma = 0.4$ is shown in Figure 5.

5.2. Algorithm Performance. We determine the cost upper bound and learning parameters through the above experiments. Subsequently, we will examine the performance of the TSNS algorithm in terms of user QoE and load balancing by comparing it with classical methods and related work.

(i) Improved ε-greedy: the edge node with the highest utility value is explored or selected with a certain probability. After the algorithm is improved, ε keeps getting smaller, and the exploration probability keeps decreasing as the number of selections increases

(ii) UCB: all optional but not yet selected edge servers are first explored. Subsequently, the edge node with the largest utility value is selected, and the utility value is updated after each selection

(iii) UBL [40]: based on the improvement of the general greedy algorithm, the utility value of the selected edge node is updated after each selection. If the same edge server is selected twice in a row, the utility value of the corresponding server is updated to a temporary value

(iv) LBCO [35]: first, determine the number of mobile devices offloaded to each edge node, consider the different available uplink data rates of the user-side devices and the computing power of the edge nodes, calculate the upload and service times for each task, obtain the set of edge nodes available to the users, calculate the corresponding times, and force each user to select the optimal edge node for the task

(v) MTOTC [27]: each user has partially selectable edge nodes, and the selection probability of all selectable nodes is summed to 1. The stochastic congestion game with incomplete information is performed based on the careful consideration of each user's task type and different task volumes. When the probability of all users selecting an edge node is 1 or the probability within an acceptable error range is greater than the set value, the game stops, and the corresponding edge node is the final choice of users

We evaluated the methodology from two main perspectives.

(i) QoE: the metric is expressed in terms of the average reward earned by users after uninstalling a task and is necessary for measuring service quality

(ii) Load balancing: this metric compares the total number of tasks ultimately served by each edge service but specifically considers the cumulative computation time for task processing in each service. This manuscript's load balancing degree is the primary metric to measure DDoS attack mitigation

In Figure 6, by computing the actual reward after each selection, we obtain a graph of the evolutionary trend of the average reward for each algorithm and, in turn, represent it as the evolutionary trend of the QoE. First, the algorithm was compared with the TS algorithm, which is based on the M/M/1 queuing model and the classical algorithms (improved ε-greedy and UCB) within the framework of the MAB algorithm. We find that the algorithm with the G/M/1 model will significantly outperform the case with the M/M/1 model in terms of QoE performance, having more significant advantages and potential. The algorithm that uses the M/M/1 model is similar to the UCB algorithm but significantly lower than the improved ε-greedy algorithm and the TSNS algorithm. Subsequently, during the comparison with related work, we found that the UBL, LBCO, and MTOTC algorithms reach their QoE peaks relatively quickly and are largely stable. In contrast, the TSNS algorithm suffers from a slow learning ascent. However, as the user assignment process continues, the TSNS algorithm outperforms the other algorithms in terms of QoE overall.

We can find that all algorithms in the MAB algorithm framework fluctuate to some extent at the operation beginning, especially during the first 100 edge user assignments. Because properties, such as the service rate of all servers, are unknown to the algorithm in the MAB framework at the beginning, the quality of user assignment could be gradually improved through continuous selection. Considering the influence of stochastic factors in the actual task arrival and processing process, server performance may also change with time; we introduce a nonstationary factor in the algorithm improvement; i.e., after each task assignment, a reward is calculated based on the task processing process, and the service rate of the edge servers is updated based on the reward.

In experiments, we count the specifics of user selection of edge nodes in different methods and represent them as Figure 7. We can see that large numbers of clusters form the representation graph for each method. The centers of the clusters represent edge servers, while the ends represent users, and the connecting lines between them represent their selection relationships. The size and density of the clusters can reflect the uniformity in selecting edge nodes by users. Among them, UBL, LBCO, and improved ε-greedy algorithms mainly focus on choosing some fixed edge nodes, and fewer edge nodes connect more users. In contrast, the TSNS, MTOTC, and UCB algorithms can distribute edge users more evenly, and the number of users served by each edge node is similar.

However, since the task volume of tasks to be processed by different users and the computational capacity of edge nodes vary, we also need to discuss the task processing of each edge node more specifically.

As in Figure 8, we count the work of each edge server between methods. Where the vertical coordinate represents the accumulated computation time of each edge server, which is expressed as the degree of load, ideally, the degree of load should be essentially similar between edge servers, although there are some fluctuations. This figure shows more intuitively that the load within the UCB, TSNS, and MTOTC algorithms are relatively homogeneous, compared to other algorithms, with slight fluctuations basically around a certain level. To quantify this balance's level more concretely, the changes in stay time are calculated and subsequently expressed as the variance. As shown in Figure 9,

we can conclude that the TSNS algorithm has some advantages in load balancing compared with other algorithms. This advantage is beneficial in resource-limited edge environments, facilitating the mitigation of DDoS attack processes and, in turn, reducing the probability of system breaches.

6. Conclusion

In this paper, to slow down the DDoS attack process in edge computing, we have focused on the EUA problem in a 5G ultradense cellular network scenario and considered improving the load balancing of edge servers while guaranteeing the QoE. To quantify the QoE, we have introduced the MAB algorithm framework and added nonstationary factors to the learning mechanism. Considering the effect of scheduling algorithms on the Markov property of the task arrival process, we have introduced the G/M/1 queueing theory model to EUA for the first time. We have focused on processing specific tasks in each edge server and conducted a series of experiments on real-world datasets, which verified the strength and potential of the algorithm in the target scenario.

In future research, in the context of non-orthogonal multiple access (NOMA) for 5G networks, we will consider more general cases of load balancing of edge demand response under the impact of latency and energy consumption. And we will slow down the process of DDoS attack in edge computing by pursuing load balancing of edge servers. First, we will specifically consider the number and performance of physical machines installed in each edge server and further consider the specific processing process after tasks reach edge servers; subsequently, we will combine cloud-edge collaboration and collaboration among edge nodes to set the threshold to determine whether users need to receive cloud services; more importantly, we will considering the performance of the algorithm in three aspects: mobile users, edge infrastructure providers, and edge service providers, considering the QoE, system energy consumption, and DDoS attack mitigation, to make the model more generalized, etc.

Data Availability

The data used to support the findings of this study is cited in the article and can be viewed via the link https://github.com/swinedge/eua-dataset.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: the communication perspective," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322–2358, 2017.

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

[3] P. Mach and Z. Becvar, "Mobile edge computing: a survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[4] W. Z. Khan, E. Ahmed, S. Hakak, I. Yaqoob, and A. Ahmed, "Edge computing: a survey," *Future Generation Computer Systems*, vol. 97, pp. 219–235, 2019.

[5] Z. Xu, G. Zou, X. Xia et al., "Distance-aware edge user allocation with QoE optimization," in *IEEE International Conference on Web Services (ICWS)*, pp. 66–74, Beijing, China, Oct 2020.

[6] X. Ge, S. Tu, G. Mao, C. Wang, and T. Han, "5G ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[7] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.:a survey and analysis of security threats and challenges," *Future Generation Computer Systems*, vol. 78, pp. 680–698, 2018.

[8] Q. He, C. Wang, G. Cui et al., "A game-theoretical approach for mitigatingedge DDoS attack," *IEEE Transactions on Dependable and Secure Computing*, 2021.

[9] O. Osanaiye, K. K. R. Choo, and M. Dlodlo, "Distributed denial of service (DDoS) resilience in cloud: review and conceptual cloud DDoS mitigation framework," *Journal of Network and Computer Applications*, vol. 67, pp. 147–165, 2016.

[10] G. Cui, Q. He, X. Xia et al., "Demand response in NOMA-based mobile edge computing: a two-phase game-theoretical approach," *IEEE Transactions on Mobile Computing*, 2021.

[11] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4377–4387, 2019.

[12] W. Z. Zhang, I. A. Elgendy, M. Hammad et al., "Secure and optimized load balancing for multitier IoT and edge-cloud computing systems," *IEEE Internet of Things Journal*, vol. 8, no. 10, pp. 8119–8132, 2021.

[13] P. Dai, Z. Hang, K. Liu et al., "Multi-armed bandit learning for computation-intensive services in MEC-empowered vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 7821–7834, 2020.

[14] F. Zhang and M. M. Wang, "Stochastic congestion game for load balancing in mobile-edge computing," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 778–790, 2021.

[15] P. Zhao, H. Tian, K. Chen, S. Fan, and G. Nie, "Context-aware TDD configuration and resource allocation for mobile edge computing," *IEEE Transactions on Communications*, vol. 68, no. 2, pp. 1118–1131, 2020.

[16] S. Misra, S. P. Rachuri, P. K. Deb, and A. Mukherjee, "Multi-armed-bandit-based decentralized computation offloading in fog-enabled IoT," *IEEE Internet of Things Journal*, vol. 8, no. 12, pp. 10010–10017, 2021.

[17] S. Ghoorchian and S. Maghsudi, "Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 279–293, 2021.

[18] Y. Hu, *Operational Research Course*, Tsinghua University Press, Beijing, China, 1998.

[19] U. N. Bhat, "An introduction to queueing theory: modeling and analysis in applications," *Boston, MA: Birkhäuser*, vol. 36, 2008.

[20] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of Queueing Theory*, John Wiley & Sons, Hoboken, New Jersey, U.S, 2018.

[21] P. Lai, Q. He, M. Abdelrazek et al., "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *16th International Conference on Service-Oriented Computing (ICSOC2018)*, pp. 230–245, Hangzhou, China, Nov 2018.

[22] R. S. Silva, C. C. Meixner, R. S. Guimarães et al., "REPEL: a strategic approach for defending 5G control plane from DDoS signalling attacks," *IEEE Transactions on Network and Service Management*, vol. 18, no. 3, pp. 3231–3243, 2021.

[23] Y. Li, Y. Zhao, J. Li, X. Yu, Y. Zhao, and J. Zhang, "DDoS attack mitigation based on traffic scheduling in edge computing-enabled TWDM-PON," *IEEE Access*, vol. 9, pp. 166566–166578, 2021.

[24] Z. Liu, X. Yin, and Y. Hu, "CPSS LR-DDoS detection and defense in edge computing utilizing DCNN Q-learning," *IEEE Access*, vol. 8, pp. 42120–42130, 2020.

[25] S. Yu, J. Zhang, J. Liu, X. Zhang, Y. Li, and T. Xu, "A cooperative DDoS attack detection scheme based on entropy and ensemble learning in SDN," *EURASIP Journal on Wireless Communications and Networking*, vol. 2021, 21 pages, 2021.

[26] V. Rajasekar, B. Predić, M. Saracevic et al., "Enhanced multimodal biometric recognition approach for smart cities based on an optimized fuzzy genetic algorithm," *Scientific Reports*, vol. 12, pp. 1–11, 2022.

[27] M. H. Saračević, S. Z. Adamović, V. A. Miškovic et al., "Data encryption for Internet of Things applications based on Catalan objects and two combinatorial structures," *IEEE Transactions on Reliability*, vol. 70, no. 2, pp. 819–830, 2021.

[28] V. Di Valerio and F. Lo Presti, "Optimal virtual machines allocation in mobile femto-cloud computing: an MDP approach," in *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp. 7–11, Istanbul, Turkey, April 2014.

[29] M. G. R. Alam, M. M. Hassan, M. Z. Uddin, A. Almogren, and G. Fortino, "Autonomic computation offloading in mobile edge for IoT applications," *Future Generation Computer Systems*, vol. 90, pp. 149–157, 2019.

[30] X. Qiu, L. Liu, W. Chen, Z. Hong, and Z. Zheng, "Online deep reinforcement learning for computation offloading in blockchain-empowered mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 8, pp. 8050–8062, 2019.

[31] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *IEEE International Symposium on Information Theory (ISIT)*, pp. 1451–1455, Barcelona, Spain, July 2016.

[32] K. Zhang, Y. Mao, S. Leng, S. Maharjan, and Y. Zhang, "Optimal delay constrained offloading for vehicular edge computing networks," in *IEEE International Conference on Communications (ICC)*, pp. 1–6, Paris, France, May 2017.

[33] S. M. Shahrear Tanzil, O. N. Gharehshiran, and V. Krishnamurthy, "Femto-cloud formation: a coalitional game-theoretic approach," in *IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, San Diego, CA, USA, Dec 2015.

[34] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: a load-balancing solution," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 2, pp. 2092–2104, 2020.

[35] 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 Transactions on Vehicular Technology*, vol. 66, no. 8, pp. 7432–7445, 2017.

[36] X. Wang, Y. Han, C. Wang, Q. Zhao, X. Chen, and M. Chen, "In-Edge AI: intelligentizing mobile edge computing, caching and communication by federated learning," *IEEE Network*, vol. 33, no. 5, pp. 156–165, 2019.

[37] T. Wang, Y. Liang, Y. Zhang et al., "An intelligent dynamic offloading from cloud to edge for smart IoT systems with big data," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2598–2607, 2020.

[38] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, 2019.

[39] Y. Cao, R. Ji, L. Ji, G. Lei, H. Wang, and X. Shao, "MPTCP: a learning-driven latency-aware multipath transport scheme for industrial internet applications," *IEEE Transactions on Industrial Informatics*, 2022.

[40] T. Lattimore, *Bandit Algorithms*, Cambridge University Press, Cambridge, United Kingdom, 2020.

[41] M. Kamoun, W. Labidi, and M. Sarkiss, "Joint resource allocation and offloading strategies in cloud enabled cellular networks," in *IEEE International Conference on Communications (ICC)*, pp. 5529–5534, London, UK, June 2015.

[42] W. Labidi, M. Sarkiss, and M. Kamoun, "Energy-optimal resource scheduling and computation offloading in small cell networks," in *International Conference on Telecommunications (ICT)*, pp. 313–318, Sydney, NSW, Australia, April 2015.

[43] A. Slivkins, "Introduction to multi-armed bandits," 2019, http://arxiv.org/abs/1904.07272.

[44] D. J. Russo, B. Van Roy, A. Kazerouni, I. Osband, and Z. Wen, "A tutorial on Thompson sampling," *Foundations and Trends in Machine Learning*, vol. 11, no. 1, pp. 1–96, 2017.

[45] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, Cambridge, Massachusetts, United States, 2018.

[46] M. Frigge, D. C. Hoaglin, and B. Iglewicz, "Some implementations of the boxplot," *The American Statistician*, vol. 43, no. 1, pp. 50–54, 1989.