Hindawi
Complexity
Volume 2023, Article ID 5222504, 20 pages
https://doi.org/10.1155/2023/5222504

Research Article

An Auction-Based Bid Prediction Mechanism for Fog-Cloud Offloading Using Q-Learning

Reza Besharati, Mohammad Hossein Rezvani, and Mohammad Mehdi Gilanian Sadeghi

Department of Computer and Information Technology Engineering, Qazvin Branch, Islamic Azad University, Qazvin, Iran

Correspondence should be addressed to Mohammad Hossein Rezvani; mhossein.rezvani@gmail.com

Received 8 October 2022; Revised 12 December 2022; Accepted 27 December 2022; Published 17 January 2023

Academic Editor: Roberto Natella

In the fog computing paradigm, if the computing resources of an end device are insufficient, the user's tasks can be offloaded to nearby devices or the central cloud. In addition, due to the limited energy of mobile devices, optimal offloading is crucial. The method presented in this paper is based on the auction theory, which has been used in recent studies to optimize computation offloading. We propose a bid prediction mechanism using Q-learning. Nodes participating in the auction announce a bid value to the auctioneer entity, and the node with the highest bid value is the auction winner. Then, only the winning node has the right to offload the tasks on its upstream (parent) node. The main idea behind Q-learning is that it is stateless and only considers the current state to perform an action. The evaluation results show that the bid values predicted by the Q-learning method are near-optimal. On average, the proposed method consumes less energy than traditional and state-of-the-art techniques. Also, it reduces the execution time of tasks and leads to less consumption of network resources.

1. Introduction

Recently, a new paradigm called fog computing has emerged, also known as "cloud at the edge" [1]. The use of this technology in areas such as e-health [2, 3], industry [4, 5], vehicular traffic management [6–8], and agriculture [9] is expanding rapidly. The core idea of fog computing is to transfer large volumes of data from the Internet of Things (IoT) devices to the remote cloud, especially for delay-constraint applications. Here, if an end device's computing capacity is insufficient, the task is offloaded to a near fog device. Similarly, if the resources in the fog device are inadequate, the task is offloaded to the remote cloud.

A review of the literature shows that the primary concern in most fog computing research is the minimization of execution delays. This operation, if performed alone, may result in higher energy consumption throughout the system. This is because, with such a single goal, mobile devices tend to offload their tasks to the nearest fog, which does not necessarily lead to minimal total energy consumption. Therefore, performing a joint optimization of delay and energy consumption is necessary.

So far, extensive research has been conducted on offloading optimization in fog computing. Some of the essential methods used in the literature are game theory [10], auction theory [1, 11, 12], probabilistic modeling [13], heuristic [14], and metaheuristic [15]. Recently, the use of machine learning methods, especially reinforcement learning (RL) [16] to optimize offloading has received much attention from the research community. Here, each agent learns to behave in a way that leads to optimizing a predetermined goal by performing a series of actions and inspecting the reward/penalty. This learning method is inspired by behavioral psychology. In other words, the RL enables an agent to learn in an interactive environment using experiences gained from previous actions. For example, when a child's hand burns in the face of heat, he/she quickly learns that fire is dangerous. Pleasure/pain is an excellent example of the reward/penalty through which humans and animals learn environmental behavior.

We believe that the auction theory is one of the most important mathematical tools to motivate fog nodes to participate in offloading operations. We consider the bandwidth of services as a commodity. Fog nodes and cloud datacenter play the role of bidders and auctioneers, respectively. Each fog node has an output queue (buffer). Once the output buffer becomes full, the fog node requests the auction operation by sending a message to the upstream (parent) node. The parent node announces the winner after collecting the bids of all downstream nodes. In the proposed mechanism, nodes participating in the auction report their bid values to the corresponding parent nodes. The node with the highest bid value wins the auction and offloads its tasks to the parent node. Bid value has an important impact on the winning of a participating node. The operation is performed suboptimally if the buffer of the winner node is empty or it has no task to offload. This, in turn, leads to increased energy consumption and latencies in executing tasks. Therefore, there is a need to design an optimal auction mechanism. For this purpose, the designed mechanism must guarantee optimal bidding. In this paper, we use RL to predict bid values.

Various studies have been conducted for bid prediction using RL. The difference between these studies is in the goals and, most importantly, the subtleties used in the RL system design. Designing actions, states, and reward functions are some of the critical differences among previous researches. The main contribution of this paper is as follows:

(i) We solve the joint minimization of latency and energy consumption with the Q-learning method. Unlike other RL research, we model the system with queuing theory to better capture agents' behavior in the real world.

(ii) We design a price-based reward function in the Q-learning model. In this function, each joule of energy consumption has a predetermined price. This paper complements one of the state-of-the-art research [1], in which each agent uses a *second-price sealed-bid (SPSB)* auction mechanism for offloading. In our modeling in this paper, the user who wins the auction has a different state than the others. The winning user can immediately use the fog device resources to offload his/her tasks. Nevertheless, our proposed model considers different situations for other users who may resubmit their requests shortly.

The remainder of this paper is organized as follows: Section 2 reviews the most critical studies on offloading optimization based on RL methods; Section 3 explains the system model; Section 4 describes the proposed method; Section 5 presents the evaluation results of the proposed method along with statistical analysis; finally, Section 6 concludes the study and outlines future research trends.

2. Related Work

Machine learning is a primary method to solve the offloading problem. Significant efforts have focused on supervised learning methods [4, 17] and reinforcement learning (RL). Given that our proposed method is based on RL, we focus on this type of research in this section. The RL is a method based on rewarding desirable behaviors or punishing undesirable behaviors [16]. Each agent can understand and interpret the environment and perform several actions here. The agent's sequence of movements over time through trial and error leads to learning the optimal state. The main difference between previous research is in arranging the above details and presumptions.

We now proceed to explain research in the scope of RL. Most of the research in this area is related to deep reinforcement learning (DRL). Readers interested in studying DRL models for IoT applications can refer to [18, 19]. Santos et al. [20] proposed a DRL method for embedding service function chains (SFCs) in a fog environment using mixed integer linear programming (MILP). They aimed at joint minimization of energy consumption and cost. Then, they showed that the proposed method could achieve a 95% acceptance ratio for user requests. Gazori et al. [21] proposed a task scheduling algorithm to minimize the latency and cost of tasks. They also paid special attention to load balancing and showed that the proposed method could resolve the challenges of the single point of failure. Zhang et al. [7] targeted the problem of selecting and switching mobile network operators (MNOs) by considering the cost of switching and QoS requirements among MNOs and cloud/fog servers. Using the DRL, they proposed a switching policy that could guarantee the average long-term cost of each vehicle with reliable latency. Performance evaluations using datasets collected from a city-wide LTE network showed that their proposed approach could reduce the cost of payment by any vehicle connected to the fog. Rahman et al. [22] proposed an algorithm to minimize fog radio access networks (F-RANs) latency. Their algorithm uses DRL to decide on local execution or offloading and allocates the desired amount of computing resources and power. A similar study was conducted by Jazayeri et al. [23] to jointly minimize latency and energy consumption using the DRL. Their results showed that the proposed method could reduce latency, energy consumption, and network usage compared to full local execution and First-Fit (FF) methods.

Chen et al. [24] solved the same problem using the deep deterministic policy gradient (DDPG) algorithm. The most important advantage of this method is that it does not need to know the transmission probabilities in different network states. The simulation results showed the superiority of the proposed method compared to the policy gradient (PG), deterministic policy gradient (DPG), and actor-critic (AC) methods. Baek and Kaddoum [25] proposed a combination of DRL and game theory to solve the offloading optimization problem. In this game, each fog node cooperates to maximize local rewards while it only has access to local observations. To deal with partial visibility, the DRL approximates the optimal value functions. The results show that the proposed algorithm can achieve a higher success rate and fewer overflows than the baseline methods. Shi et al. [8] aimed at minimizing execution delay using the AC-based DRL. The proposed algorithm maximizes expected rewards and policy entropy so that vehicles are motivated to share their idle computing resources through dynamic pricing. Chen et al. [5], by leveraging collaborative DRL, propose a method to minimize the delay and energy consumption of tasks. This study considers the mobility of mobile users to select the most effective samples of shared experiences. Also, they use a distributed collaborative method to learn the probability distribution of the approximate reward and optimize the parameters. The simulation results show that their method has fast convergence and high stability. The DRL method has also been used in fog computing for other purposes. Some of these goals include caching [26], combining blockchain with fog computing [2], and cloud-fog optical networks [27]. Readers interested in studying other DRL-based research can refer to [28, 29].

We now review research conducted using other RL methods. Mebrek et al. [30] targeted the problem of joint computation and communication optimization through game theory and RL. Their goal is to achieve an optimal policy that makes a trade-off between energy consumption and QoS. The core idea of the proposed method is to learn the optimal offloading policy without having prior knowledge of system dynamics. Cui et al. [31] proposed an optimal offloading mechanism based on the multiarmed bandit (MAB) theory for unmanned surface vehicles (USVs). Their sole purpose is to minimize offloading delays in environments where the topology and wireless channel states constantly change. Dehury and Srirama [32] conducted a similar study to provide personalized services. Neelakantam et al. [33] designed an offloading system using Q-learning to minimize passenger waiting time in a smart city. They showed that the Q-learning method is more effective than SARSA's popular RL method. Nassar and Yilmaz [34] targeted the problem of latency minimization in the F-RAN for 5G communications. They compared the performance of several RL methods, including SARSA and expected SARSA, with the baseline method. Their results show that RL methods consistently achieve the best possible performance regardless of the IoT environment.

So far, very few studies have been conducted regarding bid value prediction in online auctions using RL. For example, in [35], the stochastic game predicts the bid value. The authors then solved the problem with multiagent RL. In [36], the environment depends on the number of applied impressions. Therefore, the proposed solution is not suitable for dynamic and online environments. In [37], the Markov decision process (MDP) method was used to hold an online auction. The authors first model the bid prediction problem with MDP and define transition and reward functions based on users' click rates. They then solve the problem by leveraging dynamic programming. In [38], CMDP optimizes the bid value in a sequential auction. The authors model the CMDP with a six-element tuple. Tuple elements are state, action, state transition function, reward, cost, and constraint value, respectively. They consider click-through rate (CTR) as the state and the number of clicks on each ad as the reward. After modeling the CMDP, they solve the problem by leveraging linear programming.

RL has also been used in other areas of fog computing. Due to space limitations, we do not review this research in depth. The most interesting topics that have attracted the attention of researchers are traffic signal controlling [39], secure offloading [40], medical information processing [41], load balancing [42, 43], and service function chain allocation [44].

After reviewing the literature, the following points can be presented as the most important differences between our modeling and previous research:

(1) None of the above research has used queuing theory to model user behavior. Such modeling allows us to have closed-form relations for delay and energy. This, in turn, can be used later in the design of the reward function. In other words, the system model formulation based on queuing theory provides a more realistic description of the fog-cloud environment.

(2) None of the above research has designed the reward function based on price. We set a predetermined price for each joule of energy consumption. In addition to bidding values, these prices are used in the reward function, which will be formulated later in (32).

3. System Model

An illustration of the system model is shown in Figure 1. The system includes M mobile users, F fog devices, and a central cloud. Each mobile user can only connect to a fog node via the base station (BS). The set of mobile users is denoted by $U = \{u_1, u_2, \ldots, u_M\}$. Also, the set of fog nodes is denoted by $F = \{f_1, f_2, \ldots, f_F\}$. Every mobile user has several tasks to be performed. These tasks are entered into the system continuously. If the user's computational resources are insufficient to perform the task or the user intends to save his/her resources, he/she can offload them into a fog node. Similarly, if the computational resources of the fog node are insufficient, the task is offloaded to the central cloud. For each fog node f_j, we denote all users who are offloading to it by a set $C(f_j)$. For example, in Figure 1, $C(f_1) = \{u_1, u_2, u_3, u_4\}$ indicates that users u_1, u_2, u_3, and u_4 are offloading their tasks to the node f_1. Similarly, in this figure, we can write $C(f_2) = \{u_5, u_6\}$ and $C(c_1) = \{f_1, f_2\}$. The notations used in this study are shown in Table 1. We model mobile user traffic in this study as an M/M/1/K queue. As mentioned in [1], this model can capture the behavior of mobile users better than M/M/1, which was previously used by researchers [45]. Also, we model each fog node as an M/M/c/K queue [1, 46].

3.1. *The Edge Layer Modeling.*

As with previous research [1], we assume that each mobile user $u_i \in U$ can have multiple tasks to be performed. The number of these tasks follows the Poisson distribution [46] with the mean arrival rate λ_{u_i}. The size of each task submitted to the user u_i is denoted by θ_i. Performing this task requires spending several processor cycles for each bit that is denoted by σ_i. The time needed to complete the task on the processor of the mobile device u_i, hereafter referred to as local execution, is calculated as follows [47]:

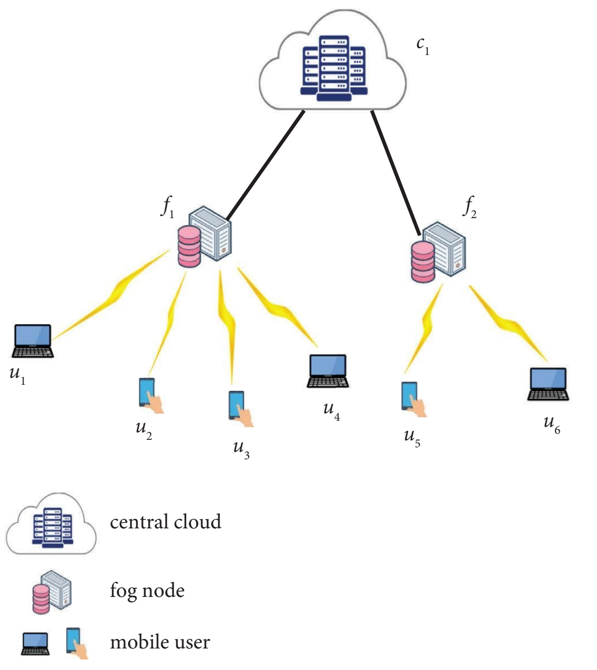


Figure 1: An illustration of the system model.

$$t^s_{u_i} = \frac{\sigma_i.\theta_i}{C_{u_i}}. \tag{1}$$

In the above formula, C_{u_i} represents the computing capacity of the mobile user u_i in cycles/sec. Therefore, the service rate on the mobile device u_i is calculated as follows:

$$\mu_{u_i} = \frac{1}{t^s_{u_i}}. \tag{2}$$

As mentioned earlier, we use an M/M/1/K queuing system to model the mobile user in this study. In this model, the maximum number of buffer rooms is K. If the number of tasks entered is more than K, the extra ones will be dropped. Tasks are entered at a rate λ_{u_i} while the acceptance rate by the mobile device is $\lambda^a_{u_i}$. Let $\pi^K_{u_i}$ represent the task blocking ratio. This means that when the mobile device queue is full, only $1 - \pi^K_{u_i}$ percent of the tasks can be served, and the rest are not accepted [48]. For admitted tasks, the local acceptance rate, $\lambda^a_{u_i}$, is as follows:

$$\lambda^a_{u_i} = \left(1 - \pi^K_{u_i}\right).\lambda_{u_i}. \tag{3}$$

Tasks that are not accepted on the mobile device must be offloaded to the appropriate fog device. Before we proceed, let us define the utilization of a mobile user u_i as follows:

$$\rho'_{u_i} = \frac{\lambda^a_{u_i}}{\mu_{u_i}}. \tag{4}$$

Given that the above birth-death process is a discrete-time Markov chain (DTMC), the following two properties always hold:

$$\pi^k_{u_i} = \left(\rho_{u_i}\right)^k.\pi^0_{u_i}, \tag{5}$$

$$\sum_{k=0}^{K} \pi^k_{u_i} = 1. \tag{6}$$

Simplifying (6) and replacing it in (5) give the blocking probability (i.e., $k = K$) as follows:

$$\pi^K_{u_i} = \frac{\left(1 - \rho_{u_i}\right).\left(\rho_{u_i}\right)^K}{1 - \left(\rho_{u_i}\right)^{K+1}}. \tag{7}$$

The average number of tasks, L_{u_i}, for the mobile user u_i is obtained as follows:

$$L_{u_i} = \sum_{k=0}^{K} k.\pi^k_{u_i} = \frac{\left(1 - \rho_{u_i}\right).\rho_{u_i}}{1 - \left(\rho_{u_i}\right)^{K+1}} \sum_{k=0}^{K} k.\left(\rho_{u_i}\right)^{k-1}. \tag{8}$$

After simplification, it follows that

$$L_{u_i} = \frac{\rho_{u_i}}{1 - \rho_{u_i}} - \frac{(K+1).\left(\rho_{u_i}\right)^{K+1}}{1 - \left(\rho_{u_i}\right)^{K+1}}. \tag{9}$$

Now, applying Little's law gives the local execution time for each task, $t^l_{u_i}$, as follows:

$$t^l_{u_i} = \frac{L_{u_i}}{\lambda^a_{u_i}} = \frac{1}{\lambda^a_{u_i}}.\left[\frac{\rho_{u_i}}{1 - \rho_{u_i}} - \frac{(K+1).\left(\rho_{u_i}\right)^{K+1}}{1 - \left(\rho_{u_i}\right)^{K+1}}\right]. \tag{10}$$

Now, the local energy consumption of the mobile user u_i is calculated as follows:

$$E^l_{u_i} = P^l_{u_i}.t^l_{u_i} = \frac{P^l_{u_i}}{\lambda^a_{u_i}}.\left[\frac{\rho_{u_i}}{1 - \rho_{u_i}} - \frac{(K+1).\left(\rho_{u_i}\right)^{K+1}}{1 - \left(\rho_{u_i}\right)^{K+1}}\right]. \tag{11}$$

In the above formula, $P^l_{u_i}$ represents the local power of the mobile user u_i. Before calculating the offloading time, $t^o_{u_i}$, let us define the offloading rate, R_i. Applying Shannon's law, the offloading rate for each mobile device u_i is obtained as follows [49]:

$$R_i = W \cdot \log_2\left(1 + \frac{P^t_i \cdot h_i}{N_0 \cdot W}\right). \tag{12}$$

In the above formula, W represents the bandwidth of the communication channel. Also, P^t_i is the transmission energy for the mobile user u_i for which $0 < P^t_i \le P^{\max}_i$. Here, $P^{\max}_i$ is the maximum transmission power of the mobile user u_i. Also, h_i represents the channel gain through which the mobile user connects to the BS, and N_0 denotes the noise that affects the channel during transmission. Now, the offloading time is calculated as follows:

$$t^o_{u_i} = \frac{\pi^K_{u_i}.\lambda_{u_i}.\theta_i}{R_i}. \tag{13}$$

Table 1: The notations used in this research.

Sym.	Symbol description
M	Total number of mobile users
F	Total number of fog devices
U	The set of all mobile users
F	The set of all fog devices
$C(f_j)$	The set of users who are offloading their tasks to the fog device f_j
λ_{u_i}	The arrival rate of tasks to a mobile user u_i
$\lambda_{u_i}^k$	The arrival rate of tasks to a mobile user u_i, when there are already k tasks in it
$\lambda_{u_i}^a$	The rate that is acceptable to a mobile user u_i
λ_{f_j}	The arrival rate of tasks to a fog node f_j
$\lambda_{f_j}^a$	The rate that is acceptable by the fog device f_j
μ_{u_i}	The rate of service for a mobile user u_i
$\mu_{u_i}^k$	The rate of service at the mobile user u_i, when there are already k tasks in it
μ_{f_j}	The rate of service at the fog device f_j
μ	The rate of service for each server inside the fog device
θ_i	The average task size entered by a mobile user u_i (bit)
σ_i	The number of CPU cycles required to perform each bit of the task in the mobile user u_i
ρ_{u_i}	The utilization of a mobile user u_i
ρ_{f_j}	The utilization of a fog device f_j
$\pi_{u_i}^k$	Probability of having k tasks in a mobile user u_i's device
$\pi_{u_i}^K$	The probability of blocking the execution of tasks for a mobile user u_i
$\pi_{u_i}^0$	Probability of not submitting any task to the mobile user u_i
$\pi_{f_j}^k$	Probability of having k tasks in a fog device f_j
$\pi_{f_j}^{K'}$	The probability of blocking the execution of tasks for a fog device f_j
$\pi_{f_j}^0$	Probability of not submitting any task to the fog device f_j
C_{u_i}	Computing capacity in a mobile user u_i (cycle/sec)
C_{f_j}	Computing capacity at the fog device f_j (cycles/sec)
$t_{u_i}^l$	Response time for each task when it is executed locally by the mobile user u_i
$t_{f_j}^l$	Response time for each task when it is executed by a fog device f_j
$t_{u_i}^o$	The time required for each task to be offloaded by the mobile user u_i toward a fog device
$t_{f_j}^o$	The time required for each task to be offloaded by the fog device f_j toward the central cloud
$t_{u_i}^s$	The time required for a task to be executed using the computing capacity of a mobile user u_i
$t_{f_j}^s$	The time required for a task to be executed using the computing capacity of a fog device f_j
L_{u_i}	The average number of tasks by the mobile user u_i
L_{f_j}	Number of tasks queued in the fog device f_j
r_i	The rate at which tasks are offloaded from the mobile device u_i to the relevant fog device
R_i	The uplink rate at which tasks are offloaded from the mobile device u_i to the relevant fog device
h_i	The gain of the channel located between the mobile user u_i and the base station (BS)
W	The channel bandwidth
N_0	The noise of the channel between the mobile user and the fog device
$P_{u_i}^l$	Power required to execute a task locally at the mobile user u_i (watts)
$P_{f_j}^l$	Power required to execute a task locally at the fog device f_j (watts)
$P_{u_i}^o$	Power required to offload a task from the mobile user u_i to the relevant fog device (watts)
$P_{f_j}^o$	Power required to transfer a task from the fog device f_j to the central cloud (watts)
P_i^t	The transmission power of the mobile user u_i
$P_i^{\max}$	The maximum transmission power of the mobile user u_i
$E_{u_i}^l$	The amount of energy consumed by the mobile user u_i to perform tasks locally
$E_{u_i}^{rem}$	The remaining energy in the mobile user u_i (J)
$E_{u_i}^{Thr}$	The threshold limit for the lowest energy level in the mobile user u_i (J)
$E_{f_j}^l$	The amount of energy consumed by the fog device f_j to perform tasks locally

Table 1: Continued.

Sym.	Symbol description
$E^{rem}_{f_j}$	The remaining energy in the fog device f_j (J)
$E^{Thr}_{f_j}$	The threshold limit for the lowest energy level in the fog device f_j (J)
$E^{o}_{u_i}$	The amount of energy consumed by the mobile user u_i to offload a task
$E^{o}_{f_j}$	The amount of energy consumed by the fog device f_j to offload a task
p_{u_j}	Price per joule of energy to be paid by the mobile user u_i (US\$/J)
$b^{f_j}_{u_i}$	The bid price per joule of energy to be paid by the mobile user u_i to the fog device f_j (US\$/J)
b_{u_i}	The budget of the mobile user u_i (US\$)
N_{f_j}	The number of end devices connected to a fog device f_j
m_{u_i}	The expected payoff of an end device $u_i \in U$ when it reports $b^{f_j}_{u_i}$ to the parent fog device f_j, while other buyers tell the truth
q_{u_i}	The probability that an end device $u_i \in U$ gets the resource when reporting its bid value b_{u_i}
S	The set of all possible states
s^i_t	The state of the mobile user u_i at a time t
S_t	The system state vector at a time t
T	The set of terminal states
A	The set of actions
a^i_t	The action performed by the mobile user u_i at a time t
r^i_{t+1}	The reward that an agent u_i gets when moved to a new state at a time $t+1$
Q	The learned action-value function
$\mathbf{Q}_*$	The optimal action-value function vector
α	The learning rate
γ	The discount factor
$V(s)$	The value function in the state s
$V_*(s)$	The optimal value function in the state s
F_{u_i}	The bidding distribution function for the mobile user u_i

Finally, the energy consumed during the task offloading from the mobile device u_i to the relevant fog device is calculated as follows:

$$E^o_{u_i} = P^o_{u_i}.t^o_{u_i} = P^o_{u_i}.\frac{\pi^K_{u_i}.\lambda_{u_i}.\theta_i}{R_i}. \tag{14}$$

3.2. The Fog Layer Modeling. Like [1], we represent the incoming traffic to each fog device with an M/M/c/K' model, in which there are c individual internal servers and a buffer of size K'. As mentioned in the previous section, if a task cannot be admitted to the mobile device $u_i \in U$, it will be offloaded to the relevant fog device f_j, for which $u_i \in C(f_j)$. The size of a task that is submitted from the mobile device u_i to the fog device f_j is denoted by θ_i. Performing this task requires spending several processor cycles for each bit that is denoted by σ_i. The time required to perform the task on the fog device f_j is calculated as follows:

$$t^s_{f_j} = \frac{\sigma_i.\theta_i}{C_{f_j}}. \tag{15}$$

In the above formula, C_{f_j} represents the computing capacity of the fog device f_j in cycles/sec. Therefore, the service rate on the fog device f_j is calculated as follows:

$$\mu_{f_j} = \frac{1}{t^s_{f_j}}. \tag{16}$$

At any given time, the fog device f_j may receive several tasks, each of which comes from the child nodes. Here, the child nodes are mobile devices that offload their extra tasks into the fog device. The rate of arrival of tasks to the fog device, λ_{f_j}, is calculated by summing the rates submitted by the children as follows:

$$\lambda_{f_j} = \sum_{i\in C(f_j)} \left(\lambda_{u_i} - \lambda^a_{u_i}\right). \tag{17}$$

Let us denote the blocking ratio and the admitted arrival rate by $\pi^{K'}_{f_j}$ and $\lambda^a_{f_j}$, respectively. Now, the utilization of the fog device f_j is defined as $\rho_{f_j} = \lambda_{f_j}/\mu_{f_j}$. Then, the probability that k tasks are present in the fog device is obtained as follows [37]:

$$\pi^k_{f_j} = \begin{cases} \dfrac{\left(\rho_{f_j}\right)^k}{k!}\pi^0_{f_j}, & 0 \le k < c, \\[2ex] \dfrac{\left(\rho_{f_j}\right)^k}{c^{k-c}c!}\pi^0_{f_j}, & c \le k < K'. \end{cases} \tag{18}$$

If we normalize the above equation according to the characteristics of DTMC, the probability of emptying the queue, $\pi^0_{f_j}$, in the fog device f_j is obtained as follows:

$$\pi^0_{f_j} = \left(\sum_{k=0}^{c-1}\frac{\left(\rho_{f_j}\right)^k}{k!} + \sum_{k=c}^{K'}\frac{\left(\rho_{f_j}\right)^k}{c^{k-c}c!}\right)^{-1}. \tag{19}$$

Suppose $a = \rho_{f_j}/c$, and then with a series of simplifications, we get the following expression [1]:

$$\pi_{f_j}^0 = \begin{cases} \left(\sum_{k=0}^{c-1} \frac{\left(\rho_{f_j}\right)^k}{k!} + \frac{\left(\rho_{f_j}\right)^c}{c!} . \frac{1 - a^{K' - c+1}}{1 - a} \right)^{-1}, & a \neq 1, \\ \left(\sum_{k=0}^{c-1} \frac{\left(\rho_{f_j}\right)^k}{k!} + \frac{\left(\rho_{f_j}\right)^c}{c!} . \left(K' - c + 1\right) \right)^{-1}, & a = 1. \end{cases} \quad (20)$$

We now proceed to calculate the number of tasks, $L_{f_j}^q$, in the fog device queue. Using the definition of mathematical expectation, we write

$$\begin{aligned} L_{f_j}^q &= \sum_{k=c+1}^{K'} (k-c).\pi_{f_j}^k = \sum_{k=c+1}^{K'} (k-c).\frac{\left(\rho_{f_j}\right)^k}{c^{k-c} c!}.\pi_{f_j}^0 \\ &= \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c}{c!} \sum_{k=c+1}^{K'} (k-c).\frac{\left(\rho_{f_j}\right)^{k-c}}{c^{k-c}} \\ &= \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c.a}{c!} \sum_{k=c+1}^{K'} (k-c).a^{k-c-1} \\ &= \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c.a}{c!} \sum_{i=1}^{K'-c} i.a^{i-1} \\ &= \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c.a}{c!} \frac{\partial}{\partial a} \left(\sum_{i=0}^{K'-c} a^i \right) \\ &= \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c.a}{c!} \frac{\partial}{\partial a} \left(\frac{1 - a^{K'-c+1}}{1-a} \right). \end{aligned} \quad (21)$$

After a series of simplifications, it follows

$$L_{f_j}^q = \frac{\pi_{f_j}^0.\left(\rho_{f_j}\right)^c.a}{c!\,(1-a)^2} \left[1 - a^{K'-c+1} - (1-a).\left(K' - c + 1\right).a^{K'-c} \right]. \quad (22)$$

The rate at which tasks arrive at the fog device is calculated when the system is not full. Therefore, it follows that

$$\lambda_{f_k}^a = \lambda_{f_k}.\left(1 - \pi_{f_j}^{K'}\right) = \mu_{f_k}.c. \quad (23)$$

On the other hand, the number of system tasks, L_{f_j}, is equal to the number of tasks in the queue, $L_{f_j}^q$, plus the number of those currently being served by CPUs. This leads us to the following relation:

$$L_{f_j} = L_{f_j}^q + c. \quad (24)$$

Altogether, from (23) and (24), it follows that

$$L_{f_j} = L_{f_j}^q + \rho_{f_k}.\left(1 - \pi_{f_j}^{K'}\right). \quad (25)$$

Similar to (10), using Little's law gives the local execution time for each task in the fog device f_j as follows:

$$t_{f_j}^l = \frac{L_{f_j}}{\lambda_{f_j}^a}. \quad (26)$$

Now, the local energy consumption of the fog device f_j is calculated as follows:

$$E_{f_j}^l = P_{f_j}^l.t_{f_j}^l = P_{f_j}^l.\left(t_{f_j}^q + \frac{1}{\mu_{f_j}} \right). \quad (27)$$

In the above formula, $P_{f_j}^l$ represents the local power of the fog device f_j. If the computing resources in the fog device are insufficient to process a task, it is offloaded to the central cloud. We denote the communication capacity between the fog device f_j and the central cloud by R_j. Similar to (13), the offloading time by the fog device f_j is calculated as follows:

$$t_{f_j}^o = \frac{\pi_{f_j}^{K'}.\lambda_{f_j}.\theta_i}{R_j}, \quad i \in C\left(f_j\right). \quad (28)$$

Finally, the energy consumed during the task offloading from the fog device f_j to the central cloud is calculated as follows:

$$\begin{aligned} E_{f_j}^o &= P_{f_j}^o.t_{f_j}^o \\ &= P_{f_j}^o.\frac{\pi_{f_j}^{K'}.\lambda f_j.\theta_i}{R_j}, \ i \in C\left(f_j\right). \end{aligned} \quad (29)$$

4. Proposed Method

In our previous research [1], we used the auction algorithm for offloading in fog computing. Each end device has a local memory (buffer); after filling this buffer, it announces the auction request to its parent. The parent node requests bid values from its children. A child with the highest bid wins the auction and offloads its tasks to its parent (higher layer). Figure 2 and Table 2 show the sequence diagram (workflow) and the messages' specifications of the proposed auction method.

The bid value's effect on the participant's winning node is very high. Since the winner performs the offloading, the bid value significantly impacts the execution time and energy consumption. An RL method has been used to predict the bid value in the auction operation.

To predict the bid value, we use one of the model-free algorithms of RL theory called Q-learning. It lies in the finite Markov decision process (FMDP) family of approaches. Here, "Q" refers to the function that the algorithm calculates. This function also called a reward function, is the expected reward, r_{t+1}, for an action a_t performed in a given state s_t. Agents learn to treat the environment based on previous

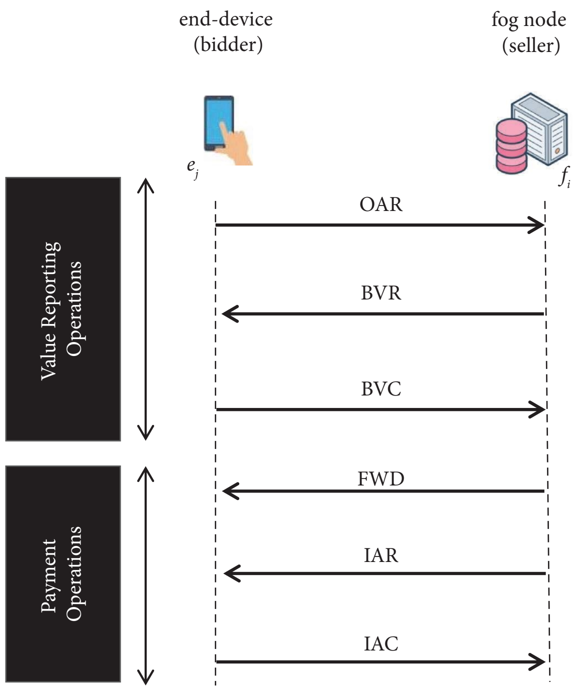


FIGURE 2: The sequence diagram (workflow) of messages.

TABLE 2: The specification of messages in the proposed mechanism.

Message	Abbr.	Message format	Description
Offloading Auction Request	OAR	⟨????????⟩	Specification of the requested source by a node e_j
Bidding Value Request	BVR	<>	The monopolist auctioneer f_i asks each of his children to send him their bid values
Bidding Value Confirm	BVC	$\langle x_{e_j} \rangle$	Each bidder e_j submits its bid value x_{e_j} to the relevant monopoly auctioneer f_i
Finding Winner Done	FWD	$\langle w \rangle$	The bidder w is introduced as the winner of the auction
Invoice Approval Request	IAR	$m_w(x_w)$	The amount of payment that the bidder w (as the auction winner) must pay to the seller f_i
Invoice Approval Confirm	IAC	< >	The winning bidder w confirms the payment $m_w(x_w)$ to the seller f_i

experiences they have gained. These previous experiences are captured by the Q value and controlled by the reward function. As shown in Figure 3, after each action a_t, the agent is moved from the state s_t to a new state s_{t+1} and receives a reward r_{t+1}. This reward reflects the importance of the action and can be the basis for deciding what to do next. The sequences of states in which each agent enters over time can be represented by $(s_0, r_0, a_0, s_1, r_1, a_1, s_2, r_2, a_2, \ldots, s_t, r_t, a_t)$. In Figure 3, the environment for each fog node f_i consists of the rest of the nodes, denoted by f_{-i}. In this way, the model captures the experiences acquired from other agents. As we will see later, the model rewards/punishes the agent for these experiences. This makes the RL process more dynamic than the supervised learning case.

Q-learning is called model-free because it does not need any prior knowledge of the environment to learn the value of an action in a given state. It can solve the bid prediction problem with rewards without needing consistency. Here, an optimal policy can maximize the expected value of the total reward through multiple iterations. In this way, Q-training can find the optimal action-selection policy for each FMDP over time through a somewhat random policy.

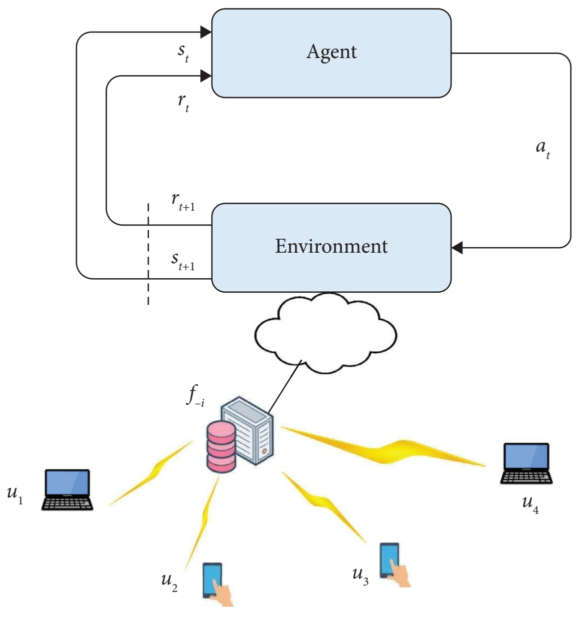


FIGURE 3: Elements of Q-learning: environment, action, state, and reward.

The components of our RL-based bid prediction approach are as follows:

4.1. Agent. For each fog device f_j, all mobile users u_i, which are $i \in C(f_j)$, are the operating agents in the proposed modeling.

4.2. State. The state of the mobile user u_i at a time t is denoted by $s_t^i \in S$, in which S represents the set of all possible states as follows:

$$S = \left\{ "\text{offload\_pending}", "\text{local\_pending}", "\text{running}" \right\}. \tag{30}$$

A "local_pending" state means that the user intends to perform the task locally on the end device. A "offload_pending" state means that the user intends to offload the task on the fog device to save local resources. Here, we use the second-price sealed-bid (SPSB) mechanism proposed in one of the state-of-the-art researches [1]. Here, first, the tasks of all users who want to offload are submitted to the fog device. Then, using the SPSB auction mechanism, they send their bids to the fog device. These are the prices that different users are willing to pay for each joule of energy consumption to the fog device. In other words, the unit of the bid price is the dollar per joule (US$/J). The offloading optimization problem for an end device $u_i \in U$ using SPSB is defined as follows [1]:

$$\max E\left[m_{u_i}\left(b_{u_i}\right)\right], \tag{31}$$

subject to

(I) The mechanism is *incentive compatible (IC)*, i.e., the requirement that q_{u_j} be nondecreasing.

(II) The mechanism is *individual rational (IR)*, i.e., the requirement that $m_{u_i}(0) \leq 0$.

In the above equation, $m_{u_i}(b_{u_i})$ denotes the expected payoff of the end device $u_i \in U$ when it reports $b_{u_i}^{f_j}$ to the parent fog device f_j, while other buyers tell the truth. Also, $q_{u_i}(b_{u_i})$ denotes the probability that the end device $u_i \in U$ gets the resource when reporting its bid value b_{u_i}.

After determining the winning user, whom we denoted by u_w, the fog device allocates computing resources to him/her. At this moment, the user state changes from "offload_pending" to "running." This means that the fog device processes the winning user's task in a non-preemptive way until it is completed. Let us denote all users except u_w, who are losers in the auction, by e_{-w}. These users may experience one of two situations: (a) the user remains in the same state "offload_pending" to participate in the next round of the auction or (b) the user refuses to offload, in which case, his/her state changes from "offload_pending" to "local_pending." Thus, if the number of end devices connected to a fog device f_j is N_{f_j}, the state of all users at a time t can be represented by a N_{f_j}-size vector $\mathbf{S}_t$. For example, suppose four users are connected to a fog node and all, but the last one intends to offload. Now, if the third user wins the auction, the system state vector is represented as follows:

$\mathbf{S}_t$ = ("offload_pending,""offload_pending,""running," "local_pending"). Now, if the capacity of the fog node is still sufficient, users whose state is "offload_pending" can try again to offload their tasks. For example, if at the next time $t+1$, the second user wins the auction, the state vector changes to $\mathbf{S}_{t+1}$ = ("offload_pending,""running,""running,""local_pending").
As is common in FMDP, at least one of the states must be defined as the terminal state T. For a user u_i, if his/her state at the time t, namely s_t^i, belongs to the set T, then its value will not change with any future action. In our problem, the state "running" is terminal, i.e., $T = \{\text{"running"}\}$. In other words, if a user's task is in "running" state (that is $s_t^i \in T$), it stays in that state until the task is completed.

4.3. Action. The action performed by the mobile user u_i at a time t is denoted by $a_t^i \in A$. Here, A is a set of all possible actions as follows:

$$A = \{ "\text{local\_request}", "\text{offload\_request}" \}. \tag{32}$$

The simplest way is to select an action based on a greedy strategy. In this strategy, the action is selected to have the highest value, $a_t^i \doteq \arg\max_{a^i} Q_t(s_t^i, a^i)$.

4.4. Transition Function. It is the probability that an agent u_i will move from the state $s_t^i = s$ to a new state $s_{t+1}^i = s'$ when it acts $a_t^i = a$. It is denoted by a matrix $P(s', r|s, a)$, which P: $S \times A \times S \longrightarrow [0, 1]$ and $\sum_{s'} \sum_r P(s', r|s, a) = 1$.

4.5. Reward Function. Depending on what action the agent chooses, it will receive a reward as follows:

$$r_{t+1}^i = \begin{cases} -p_{u_i}.E_{u_i}^l, & \text{if } s_{t+1}^i == \text{local\_pending}, \\ p_{u_i}.E_{u_i}^l - p_{u_i}.E_{u_i}^o - b_{u_i}^{f_j}.E_{f_j}^l, & \text{else if } s_{t+1}^i == \text{running}, \\ 0, & \text{else if } s_{t+1}^i == \text{offloading\_pending}. \end{cases} \tag{33}$$

As stated in the above relation, if after acting a_t^i, the mobile user u_i is moved to a new state local_pending, he/she will receive a reward $-p_{u_i}.E_{u_i}^l$. Note that the system's goal here is to minimize the energy consumption of mobile devices. In other words, the proposed algorithm encourages users to offload their tasks to fog devices instead of performing them locally on their own devices. If the user performs the task locally, he/she will receive a penalty $p_{u_i}.E_{u_i}^l$. Simply speaking, if the user offloaded the task instead of running it locally, he/she could save $p_{u_i}.E_{u_i}^l$ dollars by not consuming valuable local resources. The second condition in the above relation states that if the user is moved to a state "running", he/she will receive a reward $p_{u_i}.E_{u_i}^l - p_{u_i}.E_{u_i}^o - b_{u_i}^{f_j}.E_{f_j}^l$. Here, the first term, $p_{u_i}.E_{u_i}^l$, is the amount of money the user has saved by not performing the task locally. Simply put, if the user wanted to run the task locally instead of offloading it, he/she would have to pay $p_{u_i}.E_{u_i}^l$ dollars for local resource usage. The second term, $-p_{u_i}.E_{u_i}^o$, is the amount of money that must be paid to transmit the task to the BS. The third term, $-b_{u_i}^{f_j}.E_{f_j}^l$, is the amount of money that must be paid to the fog device to perform the task. Note that $b_{u_i}^{f_j}$ is the bid price previously offered by the user u_i to the fog device f_j. As mentioned earlier, in this research we adopt the SPSB auction mechanism proposed by Besharati et al. [1]. When all users submit their bids to the fog device, only one of them who has offered the highest bid (here, the user u_i), will win the auction. Then, the winning user u_i has to pay $b_{u_i}^{f_j}$ dollars for each joule of energy consumption to the fog device. Finally, the last condition in (33) states that if the user is moved to a state "offloading_pending", he/she will not receive any reward. Although this user tends to offload the task, his/her bid price has not been the highest compared to other competitors. Therefore, the user is temporarily moved to the state "offloading_pending" so that he/she may win the auction in the next time slot.

Based on performing a specific action, the agent is transferred from one state to another. The basic form of the Q-learning algorithm for the agent u_i is $(s_t^i, a_t^i, r_{t+1}^i, s_{t+1}^i, a_{t+1}^i)$, meaning that the agent was in the state s_t^i, performed the action a_t^i, received the reward r_{t+1}^i, and finally ended up in the state s_{t+1}^i, from where it decided to act a_{t+1}^i. By doing so, it provides a new iteration to update $Q(s_t^i, a_t^i)$.

The Q-learning algorithm is one of the off-policy RL methods. We adopt the simplest form of it, which is called single-step Q-learning. It is defined as follows:

$$Q\left(s_t^i, a_t^i\right) \longleftarrow Q\left(s_t^i, a_t^i\right) + \alpha\left[r_{t+1}^i + \gamma \max_{b^i} Q\left(s_{t+1}^i, b^i\right) - Q\left(s_t^i, a_t^i\right)\right]. \tag{34}$$

In the above formula, Q and α are the learned action-value function and learning rate, respectively. The α value strikes a balance between the agent's findings from the environment and what he/she has learned. Also, γ denotes the discount factor, which satisfies $0 \le \gamma \le 1$. A lower discount factor encourages the agent to take action sooner rather than postponing it indefinitely. In other words, it determines how much importance should be given to immediate rewards and future rewards. This helps us to avoid infinity as a reward. A value of $\gamma = 0$ means that more priority is given to immediate rewards, and a value of $\gamma = 1$ means that more importance is given to future rewards. In practice, the discount factor $\gamma = 0$ is never learned because it only considers immediate rewards, and the discount factor $\gamma = 1$ continues for future rewards, which may lead to infinity. Therefore, the optimal value for the discount factor, γ, is in the range (0, 1).

After determining the transition function P and the received reward r_{t+1}^i by the parent node (controller), the MDP problem can be easily solved using dynamic programming algorithms. Here, the core idea is to use the value function $V(s)$ to find the optimal action a_*^i. The optimal action in any state s_t^i is the action that brings the most reward to the agent. For this purpose, the state value function must be expressed in the following form, which is known as the Bellman equation:

$$V_*(s) = \max E\left(r_{t+1}^i + \gamma.V_*\left(s_{t+1}^i\right) | s_t^i = s, a_t^i = a\right). \tag{35}$$

The above formula after simplification can be rewritten as follows:

$$V_*(s) = \max_a \sum_{s',r} P\left(s', r \mid s, a\right)\left[r + \gamma.V_*\left(s'\right)\right]. \tag{36}$$

One of the most common ways to solve the Bellman equation is to rewrite it in the following recursive form:

$$V_{t+1}(s) = \max_a \sum_{s',r} P\left(s', r \mid s, a\right)\left[r + \gamma.V_t\left(s'\right)\right]. \tag{37}$$

Thus, the value of all states can be obtained. Given that $0 \le \gamma \le 1$, it can be proved that the Bellman equation will converge and, therefore, the solution is as follows:

$$V_*(s) = \lim_{t \longrightarrow \infty} V_t(s). \tag{38}$$

The key advantage of Q-learning over other RL methods is that it converges very quickly. The main reason is that Q can directly approximate the optimal action-value function, q_*. Here, the policy serves to determine which state-action pairs to be visited and updated. Like most RL methods, the prerequisite for convergence here is that all pairs continue to be updated. Under this assumption and other common indefinite approximation conditions in the sequence of size-step parameters, it is found that Q converges with a probability of 1 to q_*. Another advantage of Q-learning is that it does not require a specific model of the environment. In RL terminology, it is a model-free method and does not require a predetermined policy to find any optimal state-action pair.

Note that the Q-learning algorithm is executed both in the fog layer and in the cloud layer. Due to space limitations and the similarity of pseudocodes, we only show the offloading operation in the fog layer. The pseudocode of the Q-learning for the offloading problem is shown in Algorithm 1. Also, Algorithms 2 and 3 represent the pseudocode of the mobile user u_i and the fog device f_j (controller), respectively. Algorithm 3 itself calls Algorithm 4 to calculate the bid price per joule of energy consumption based on the second-price sealed-bid (SPSB) mechanism [1]. Let us now give a brief description of these algorithms. In lines 5–8 of Algorithm 2, if the energy consumption is within the allowable range and does not exceed the remaining energy of the mobile device, the task is performed locally. In this case, the state of the task, s_t^i, is changed to "local_execution" and then the user adds the revenue from the local energy consumption, $p_{u_i}.E_{u_i}^l$, to his/her current budget. In lines 10–12 of Algorithm 2, because the energy consumption has exceeded the residual energy of the mobile device, the task is offloaded to the fog device by calling Algorithm 3. In this case, the costs of offloading, $p_{u_i}.E_{u_i}^o$, and remote processing, $b_{u_i}^{f_j}.E_{f_j}^l$, are deducted from the mobile user's budget. Similarly, in lines 7–14 of Algorithm 3, if the task energy consumption exceeds the remaining energy of the fog device, it is offloaded to the remote cloud. In this case, in addition to the cost of offloading, $p_{u_i}.E_{u_i}^o$, and processing in the fog device $b_{u_i}^{f_j}.E_{f_j}^l$, the cost of offloading to the remote cloud, $p_{f_j}.E_{f_j}^o$, must also be deducted from the user's budget.

Otherwise, if the user's budget is sufficient to process the task on the fog device, the task state is changed to "offload_pending"; otherwise, it must be processed locally on the mobile device itself. In lines 15–17 of Algorithm 3, a second-price sealed-bid (SPSB) auction mechanism is held by calling Algorithm 4. Readers interested in a detailed study of Algorithm 4 can refer to [1]. Finally, after determining the auction winner, the task state for the user who won the auction is changed to "running". Then, in line 18 of Algorithm 3, the optimal action-value function vector $\mathbf{Q}_*$ is found by calling Algorithm 1. This vector specifies exactly on which device each task should be performed. The task must be performed locally on the user's device, the fog device, or the remote cloud. Here, an important question arises regarding Algorithm 4: if the bid value of two nodes is the same, which one will have more priority? According to the implementation in [1], in such a case, the node whose remaining buffer capacity and the remaining energy are less has a higher priority.

5. Performance Evaluation

In this section, after explaining the simulation settings, we will compare the proposed method with one of the state-of-the-art methods [1]. This comparison is made in terms of important criteria such as execution time, power consumption, and network usage.

5.1. Experimental Setting. The iFogSim software has been used for simulation [50]. Also, statistical analysis of the simulation results was performed using SPSS software. The results were obtained on a laptop with a 64-bit Intel® Core™ i5-8269U processor, 6 MB Cache, 4 Cores, 4.20 GHz CPU, and 8 GB of RAM. As with other research [1], a three-tier hierarchical tree structure has been used for the simulation, which includes the central cloud layer, the fog device layer, such as routers and switches, and the end device layer.

As mentioned earlier, this study adopts the second-price sealed-bid (SPSB) auction mechanism proposed in [1] to determine the price of computational resources concerning parent-child nodes. Accordingly, in the diagrams in this section, we abbreviate the baseline method of [1] to *SPSB_Auction*. We also abbreviate our proposed method to *SPSB_Auction_RL*. Note that there is no learning mechanism in *SPSB_Auction* [1], and the only contribution is the auction-based pricing method. However, *SPSB_Auction_RL* uses a dynamic approach based on RL. We also compare our results with the default offloading mechanism used in the iFogSim, i.e., *First-Come-First-Serve (FCFS)*. This method asks the corresponding parent node to perform the offloading operation when a node's queue becomes full.

For the sake of simplicity, we ignore interdependencies between tasks. In other words, each task is independent of its previous and subsequent tasks. Although this assumption is not always correct, it still makes sense in many web applications. The time between the arrival of tasks in the iFogSim simulator follows an exponential distribution in which the arrival rate, λ_{u_i}, is 50 tasks per minute. The buffer size of each fog device is 20. For fairness, we adopt the settings used by [1] on iFogSim. Table 3 shows the details of the simulation settings. Since some parameters of iFogSim follow probabilistic distributions, we performed each test 40 times. Then, average values were used to draw the curves.

5.2. Execution Time. Figure 4 shows a comparison of runtime for different numbers of fog nodes. Like most previous research [51, 52], we consider the number of fog nodes almost small. This is because a typical fog device does not have as much computing power as a central cloud. As shown in the figure, the average execution time of all methods increases as the number of nodes increases. However, it shows much less growth for *SPSB Auction_RL*. From a practical point of view, the proposed method offers an attractive implication for scalability for network designers.

An interesting point in Figure 4 is the dramatic increase in response time in the *SPSB-Auction* method (green curve). To explain this, note that the response time consists of two parts: (1) first, it takes time for the nodes to submit their bidding values; (2) after the auction winner is determined, resources are allocated based on the announced price. This is the time that was previously calculated in (15) and (28). In the *SPSB-Auction* method (green curve), the bidding price is proportional to the number of buffer rooms and the remaining energy of the nodes [1]. As time passes, the nodes experience a lack of buffer and energy. This makes them less likely to win the auction. In other words, when resources are more utilized, the time required to determine the winning node becomes longer. The advantage of the proposed method (blue curve) appears exactly here. Here, better allocation of resources by leveraging RL leads to more equitable distribution of resources. This, in turn, causes the nodes to experience less buffer and energy shortages. Nodes whose buffer is empty or having more remaining energy do not need to offload tasks. Thus, they do not participate in auction operations. By reducing the number of nodes participating in the auction, the time required to announce the winner is reduced dramatically.

We use the Analysis of Variance (ANOVA) test to compare the average execution time of the three methods *FCFS*, *SPSB_Auction*, and *SPSB_Auction_RL*. The ANOVA test compares the mean of a quantitative variable between more than two independent groups. This test is a generalized *T*-test between two independent samples with the same assumptions. Readers interested in studying more about these statistical tests can refer to [15, 53, 54]. The ANOVA test hypotheses are introduced as follows:

(H0) The average execution time of all methods is the same.

(H1) There is at least one method whose average execution time differs from others.

Given that the confidence interval is assumed to be 95% by default, the threshold value for accepting/rejecting the comparison results, called the Sig value, is 0.05. Table 4 shows the within-group and between-group values obtained from the ANOVA test regarding the execution time. As seen from the table, since $\mathrm{Sig} = 0.033 < 0.05$, the

```
     Input: The fog node identifier $f_j$, the set of states $S$, the set of actions $A$, the set of terminal states $T$
     Output: the optimal action-value function vector $\mathbf{Q}_*$
(1)  Set $t = 0$
(2)  for $i \in C(f_j)$ do
(3)    for $s_t^i \in S$ do
(4)      for $a_t^i \in A$ do
(5)        Initialize $Q(s_t^i, a_t^i)$ arbitrarily
(6)        Initialize terminal state value, $Q(T^i, .) = 0$
(7)      end for
(8)    end for
(9)  end for
(10) repeat
(11)   Initialize $\mathbf{S}_t$
(12)   repeat (for each step of the episode)
(13)     Choose $\mathbf{A}_t$ from $\mathbf{S}_t$ using policy derived from $\mathbf{Q}_t$ (e.g., ε-greedy)
(14)     Take action $\mathbf{A}_t$, observe $\mathbf{R}_{t+1}$, $\mathbf{S}_{t+1}$
(15)     $\mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t) \longleftarrow \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t) + \alpha[\mathbf{R}_{t+1} + \gamma \max_{\mathbf{B}} \mathbf{Q}(\mathbf{S}_{t+1}, \mathbf{B}) - \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t)]$ using equation (33)
(16)     $\mathbf{S}_t \longleftarrow \mathbf{S}_{t+1}$
(17)   until $\mathbf{S}_t \in T$
(18)   $t \longleftarrow t + 1$
(19) until Max_Num_Episodes
(20) $\mathbf{Q}_* \longleftarrow \mathbf{Q}(\mathbf{S}_t, \mathbf{A}_t)$ using equation (37)
(21) return $\mathbf{Q}_*$
```

Algorithm 1: Pseudocode of the Q-learning for the offloading problem.

```
     Input: The arrival rate of tasks $\lambda_{u_i}$, the task size of the mobile user $\theta_i$, the number of CPU cycles required to perform each bit of the
     task in the mobile user $\sigma_i$, computing capacity in mobile user $\overset{u_i}{C}$, the uplink rate at which tasks are offloaded $R_i$, the gain of the
     channel located between the mobile user $u_i$ and the BS $h_i$, the channel bandwidth $W$, the noise of the channel $N_0$, power required
     to execute a task locally $P_{u_i}^l$, the maximum transmission power of the mobile user $P_i^{\max}$, price per joule of energy to be paid by the
     mobile user $p_{u_j}$
     Output: local energy consumption $E_{u_i}^l$, the offloading energy consumption $E_{u_i}^o$
(1)  Calculate the response time of the task in local execution, $t_{u_i}^l$, using equation (10)
(2)  Calculate the offloading time of the task, $t_{u_i}^o$, using equation (13)
(3)  Calculate the local energy consumption, $E_{u_i}^l$, using equation (11)
(4)  Calculate the offloading energy consumption, $E_{u_i}^o$, using equation (14)
(5)  if ($E_{u_i}^l \le E_{u_i}^{rem}$ and $E_{u_i}^{rem} \ge E_{u_i}^{thr}$) then
(6)    $s_t^i \longleftarrow$ "local_execution"
(7)    Perform the task locally
(8)    $\text{budget}_{u_i} = \text{budget}_{u_i} + p_{u_i}.E_{u_i}^l$
(9)  else
(10)   Send $p_{u_i}$, $E_{u_i}^l$ and $E_{u_i}^o$ to the fog device $f_j$ (controller) by calling Algorithm 3
(11)   $\text{budget}_{u_i} = \text{budget}_{u_i} - p_{u_i}.E_{u_i}^o - b_{u_i}^{f_j}.E_{f_j}^l$
(12) end if
```

Algorithm 2: Pseudocode of a mobile user u_i in the proposed offloading method.

assumption H_0 is rejected. This means that the three methods under study have different average execution times. In Table 4, field F shows the amount of between-groups changes compared to within-group changes. To obtain the F-value for each group, the sum of the squares of the difference of its data concerning the mean is calculated. The larger F is, the higher the probability of rejecting hypothesis H_0. On the contrary, a small value of F indicates the nonsignificance of the difference between the groups. In the ANOVA test, the *df* field indicates the degree of freedom. Statistically, the degree of freedom is the sample size minus the number of parameters estimated from the data.

The pairwise comparison of runtime values for different methods obtained from the ANOVA test is shown in Table 5. As seen from the table, the execution time of the *SPSB_Auction* method is different from the other two methods and has a significant difference. In other words, the *SPSB_Auction* method has a longer execution delay than others. For the sake of simplicity of comparison, the average

```
Input: The arrival rate of tasks in the fog node $\lambda_{f_j}$, the task size of the mobile user $\theta_i$, the number of CPU cycles required to perform each bit of the task in the mobile user $\sigma_i$, computing capacity in the fog device $C_{f_j}$, the power required to execute a task at the fog device $P^l_{f_j}$, price per joule of energy to be paid by the mobile user $p_{u_j}$
Output: the optimal action-value function vector $\mathbf{Q}_*$
(1)  Calculate the response time of the task when it is executed by the fog device, $t^l_{f_j}$, using equation (26)
(2)  Calculate the offloading time of the task towards the central cloud, $t^o_{f_j}$, using equation (28)
(3)  Calculate the local energy consumption, $E^l_{f_j}$, using equation (27)
(4)  Calculate the offloading energy consumption towards the central cloud, $E^o_{f_j}$, using equation (29)
(5)  Call Algorithm 1 to get the optimal action-value function vector $\mathbf{Q}_*$
(6)  Update computing capacity in the fog device $C_{f_j}$
(7)  if ($E^l_{f_j} > E^{rem}_{f_j}$ or $E^{rem}_{f_j} < E^{Thr}_{f_j}$) then
(8)      Send $p_{f_j}$, $t^l_{f_j}$, and $t^o_{f_j}$ to the remote cloud
(9)      $\text{budget}_{u_i} = \text{budget}_{u_i} - p_{u_i}.E^o_{u_i} - b^{f_j}_{u_i}.E^l_{f_j} - p_{f_j}.E^o_{f_j}$
(10) else if ($p_{f_j}.E^l_{f_j} \leq \text{budget}_{u_i}$) then
(11)     $s^i_t \longleftarrow$ "offload_pending"
(12)     else
(13)     $s^i_t \longleftarrow$ "local_execution"
(14)     end if
(15) call Algorithm 4 to hold the second-price sealed-bid (SPSB) auction
(16) $w \longleftarrow$ the identifier of the node who is the winner of the auction
(17) $s^w_t \longleftarrow$ "running"
(18) call Algorithm 1 to calculate the optimal action-value function vector $\mathbf{Q}_*$
```

ALGORITHM 3: Pseudocode of a fog device f_j (controller) in the proposed offloading method.

```
Input: The identifier of the buyer $u_i$ (bidder node), the bidding distribution function $F_{u_i}$
Output: the identifier of the node who is the winner of the auction $w$, the bidding value which must be paid by the winner to the fog device $f_j$ for each Joules of energy consumption $b^{f_j}_w$
(1) Wait to receive the bidding value $x_{u_i}$ from all mobile users $u_i$
(2) $b^{f_j}_{u_i} \longleftarrow$ Maximum bidding value
(3) $w \longleftarrow$ the identifier of the node who is the winner of the auction
(4) Introduce the winner $w$ to other bidder nodes
(5) Delete the node $w$ from the set of future bidders
(6) Return $w$ and $b^{f_j}_w$
```

ALGORITHM 4: Pseudocode of the Second-price Sealed-bid (SPSB) auction [1].

TABLE 3: The simulation settings.

Module	CPU computing capacity (MIPS)	RAM (MB)	UP bandwidth (kbps)	Down bandwidth (kbps)
Cloud	44800	40000	10000	10000
Gateway (fog)	2800	4000	10000	10000
End device	3200	1000	10000	270

execution time of different methods is shown in Figure 5. Again, in this figure, it is clear that the execution time of the *SPSB_Auction* method is significantly different from other methods.

We use the *Mann–Whitney U* test to check the correlation between the execution times of the *SPSB_Auction* and *SPSB_Auction_RL* methods. Unlike the *T*-test, here, the sum of the scores is used instead of the mean. For this purpose, the samples are first merged and assigned ranks. The allocated sums are then calculated in each method separately. The total number of data points is 22, half belonging to the *SPSB_Auction_RL* method and the rest to the *SPSB_Auction*. First, the 22 data points are sorted in ascending order, and then they receive the smallest rank number 1, and the highest rank number 22. The result of the *Mann–Whitney U* test for execution time is shown in Table 6. As can be seen in the table, the "Sum of Ranks" values for the *SPSB_Auction_RL* and *SPSB_Auction* methods are 97 and 156, respectively. Since the "Sum of Ranks" in the *SPSB_Auction_RL* method is less than that of the *SPSB_Auction* method, its execution time is less than the *SPSB_Auction* method.

The Mann–Whitney U value, U_{MW}, for each dataset is calculated as follows [1]:

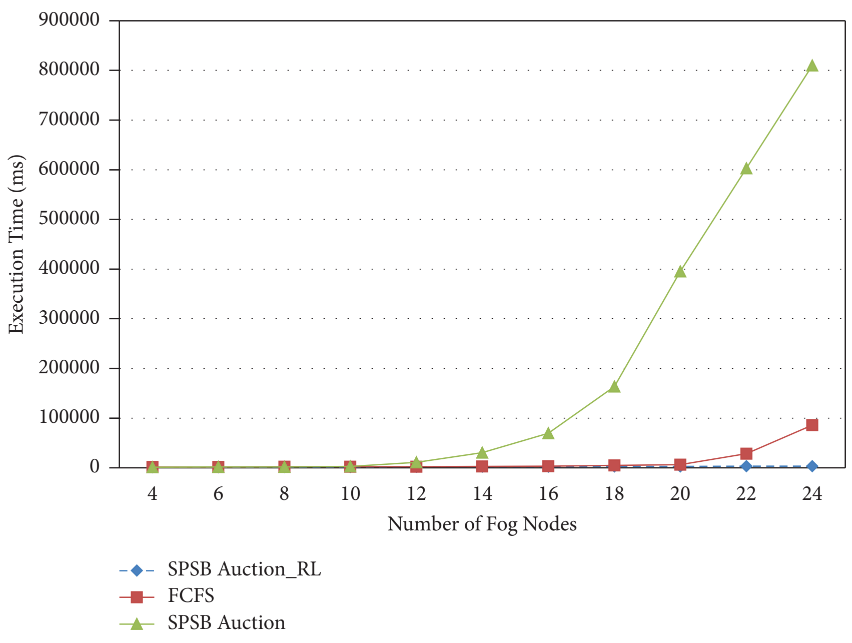


Figure 4: Execution time for different numbers of fog nodes.

Table 4: Intragroup and intergroup values obtained from the ANOVA test regarding the execution time.

ANOVA					
Execution time	Sum of squares	df	Mean square	F	Sig
Between groups	405683979302.606	2	202841989651.303	3.842	0.033
Within groups	1583952298280.72	30	52798409942.691		
Total	1989636277583.33	32			

Table 5: Pairwise comparison of runtime values for different methods obtained from the ANOVA test.

Multiple comparisons

Dependent variable: execution time

LSD

(I) Approach	(J) Approach	Mean difference (I – J)	Std. error	Sig	95% confidence interval	
					Lower bound	Upper bound
SPSB Auction_RL	SPSB auction	−240349.36364*	97978.11	0.020	−440447.3680	−40251.3593
	FCFS	−10654.45455	97978.11	0.914	−210752.4589	189443.5498
SPSB auction	SPSB Auction_RL	240349.36364*	97978.11	0.020	40251.3593	440447.3680
	FCFS	229694.90909*	97978.11	0.026	29596.9048	429792.9134
FCFS	SPSB Auction_RL	10654.45455	97978.11	0.914	−189443.5498	210752.4589
	SPSB auction	−229694.90909*	97978.11	0.026	−429792.9134	−29596.9048

*The mean difference is significant at the 0.05 level.

$$U_{MW} = S_{\text{rank}} - \frac{N(N+1)}{2}, \tag{39}$$

where N and S_{rank} denote the number of data points in each dataset and the sum of ranks, respectively. According to (39), the Mann–Whitney U values for the *SPSB_Auction_RL* and *SPSB_Auction* methods are $97–11 * 12/2 = 31$ and $156–11 * 12/2 = 90$, respectively. Figure 6 shows the largest value, 90, which corresponds to the value of Wilcoxon $W = 156$ for the *SPSB_Auction* method.

Figure 6 presents an interesting implication for the "risk aversion" property [55] of *SPSB_Auction_RL* and *SPSB_Auction* methods. Based on this property, the bid values announced by mobile users do not differ significantly from each other. This leads to the fact that the execution times of these two methods are slightly different from each other. The hypotheses used are as follows:

H_0: the selected offloading approach has a significant effect on the execution time.

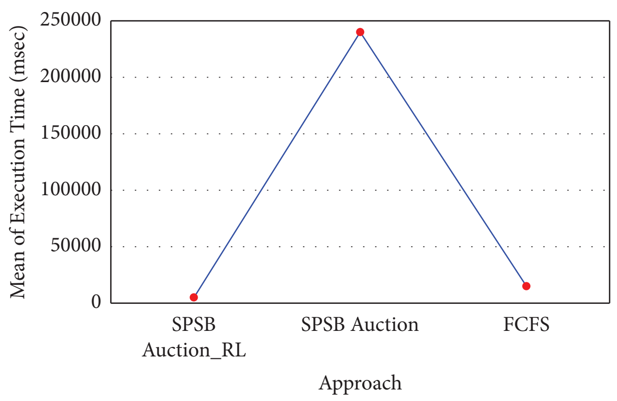
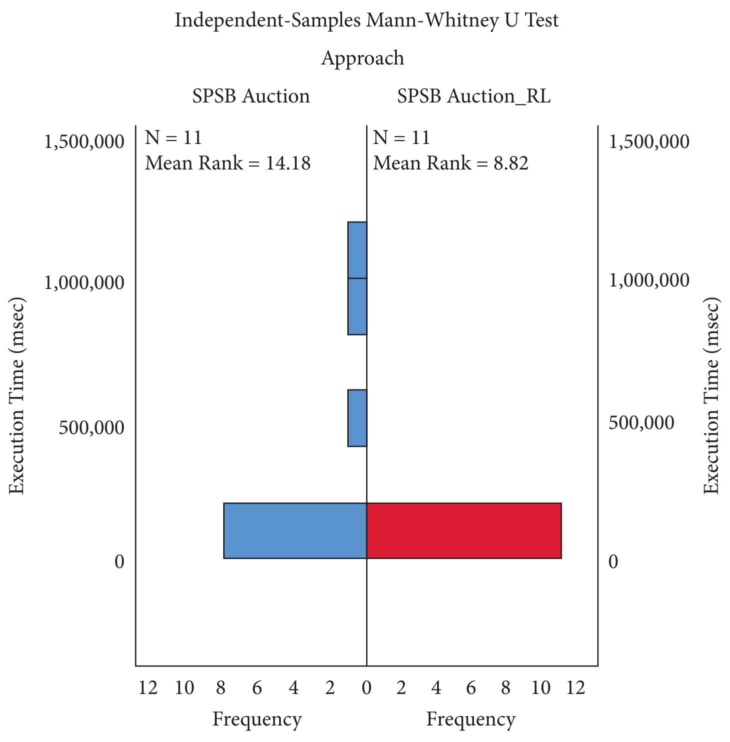
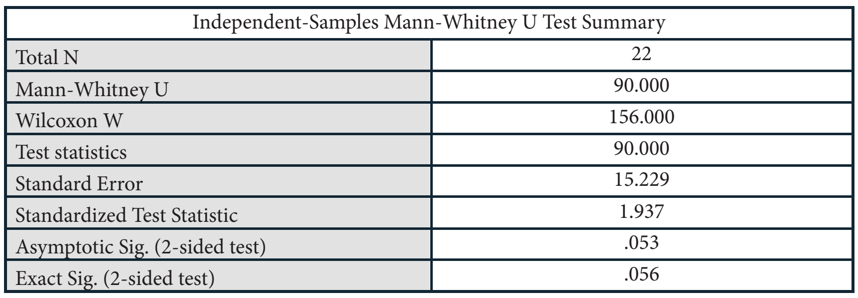


Figure 5: The average execution time for different approaches.

Table 6: Ranks assigned to each method in the Mann–Whitney *U* test.

Ranks	Approach	*N*	Mean rank	Sum of ranks
Execution time	SPSB Auction_RL	11	8.82	97.00
	SPSB auction	11	14.18	156.00
	Total	22		





Independent-Samples Mann-Whitney U Test Summary	
Total N	22
Mann-Whitney U	90.000
Wilcoxon W	156.000
Test statistics	90.000
Standard Error	15.229
Standardized Test Statistic	1.937
Asymptotic Sig. (2-sided test)	.053
Exact Sig. (2-sided test)	.056

Figure 6: Execution time analysis information of different methods in the Mann–Whitney *U* test.

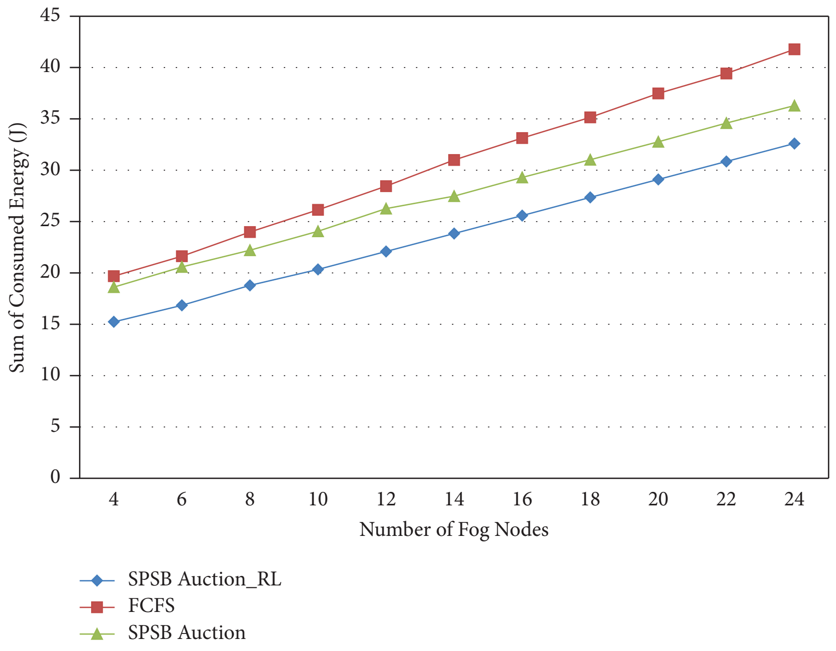


Figure 7: Energy consumption for different numbers of fog nodes.

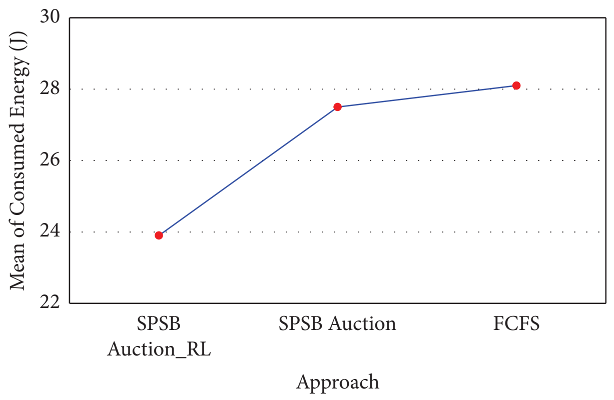


Figure 8: The average energy consumption for different approaches.

H_1: the selected offloading approach has little impact on the execution time.

Hypothesis acceptance in the Mann–Whitney U test is based on the asymptotic Sig and Mann–Whitney U value. As shown in Figure 6, we have the asymptotic Sig $= 0.053 > \alpha = 0.05$ and Mann–Whitney $U = 90$. Therefore, the hypothesis H_0 is not rejected.

5.3. Energy Consumption. Figure 7 shows the energy consumption of the entire system, which includes the total energy consumed in local execution (11) and (27) and offloading (14) and (29) in all layers. As shown in the figure, the amount of energy consumption in all methods increases with an increase in the number of fog nodes. However, the proposed method, *SPSB_Auction_RL*, manages to consume less energy than others. Due to space limitations, the results of the ANOVA test have been omitted.

For the sake of simplicity of comparison, the average energy consumption of different methods, as well as their variance, is shown in Figures 8 and 9. Again, in these figures, it is clear that the energy consumption of the *SPSB_Auction_RL* method is significantly different from other methods. Also, Figure 9 can provide an interesting implication for the stability of the proposed method. As is evident in the figure, the variance of energy consumption in the *SPSB_Auction_RL* method is less than others. This indicates that it has acceptable stability concerning selected hyperparameters, α and γ, in (34).

5.4. Network Usage. Figure 10 shows the network usage for different numbers of fog nodes. This is one of the criteria for which the iFogSim simulator generates reports. The network usage criterion is the number of bits transmitted to offload all tasks in the network. As can be seen from the figure, network usage increases as the number of fog nodes

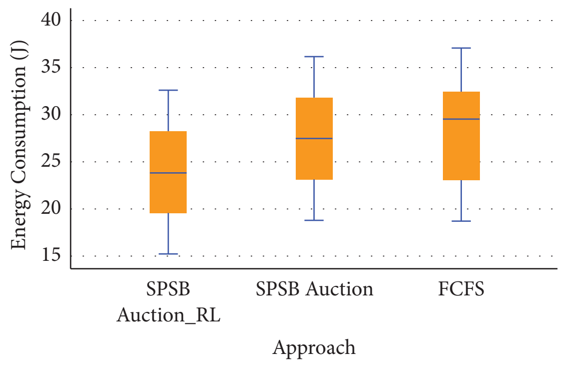


Figure 9: The variance of the energy consumption for different approaches.

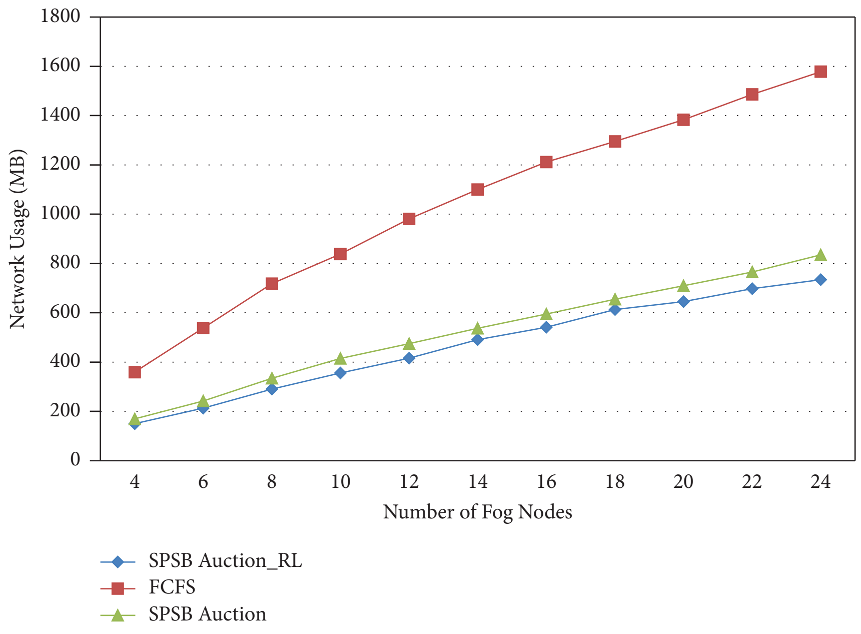


Figure 10: Network usage for different numbers of fog nodes.

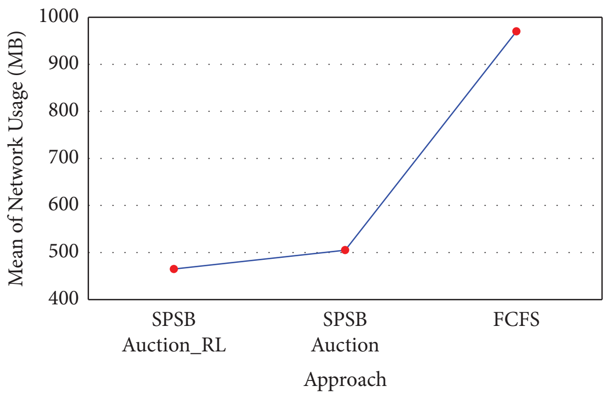


Figure 11: The average network usage for different approaches.

increases. However, the network usage in the *SPSB_Auction* and *SPSB_Auction_RL* methods is significantly less than that of the *FCFS*. Also, note that *SPSB_Auction* and *SPSB_Auction_RL* methods, themselves, have very little difference in terms of network usage.

Also, the average network usage of different methods is shown in Figure 11. Again, in this figure, it is clear that the energy consumption of the *FCFS* method is significantly different from other methods.

6. Conclusion and Future Trends

This paper used reinforcement learning to optimize bid value prediction in auction-based offloading. In our proposed mechanism, nodes participating in the auction report their bid values to their corresponding parent nodes. The node with the highest bid wins the auction and offloads its tasks to the parent node. After discussing the strengths and weaknesses of previous research, we modeled the problem by queuing theory. In this model, the response time and energy consumption of mobile users' tasks were formulated in all three layers edge, fog, and cloud. Then, using the Q-learning method, a price-based mechanism was designed to encourage users to offload their tasks so that the energy consumed by the final devices is minimized.

The performance of the proposed method was evaluated against one of the state-of-the-art methods, in which only the second-price sealed-bid auction mechanism is used for offloading. The simulation results showed that the proposed method significantly reduces task execution time compared to the baseline and the FCFS methods. The average energy consumption of the proposed method is lower than other methods. However, the FCFS and the baseline methods have almost equal energy consumption. In addition, the lower energy consumption variance makes it more stable than other methods. Unlike the FCFS method, both the proposed and baseline methods save the network resources significantly.

One future research trend is considering users' mobility and hand-off. Also, using other RL methods in combination with metaheuristic methods can probably remarkably reduce learning time.

Data Availability

The datasets generated during and analyzed during the current study are available in the data.mendeley.com/datasets/h65773tcn6/1 repository, DOI: 10.17632/h65773tcn6.1.

Consent

Informed consent was obtained from all participants included in the study.

Disclosure

This article reports the scientific findings of an academic Ph.D. thesis presented by Mr. Reza Besharati as the student and Dr. Mohammad Hossein Rezvani as the advisor. Also, Dr. Mohammad Mehdi Gilanian Sadeghi was the thesis consultant.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

All authors contributed to the study's conception and design. Simulation programming, data collection, and analysis were performed by Reza Besharati. Project navigation and checking mathematical proofs were performed by Mohammad Hossein Rezvani. Scientific consultancy and advice on the use of state-of-the-art methods for comparison with the proposed algorithm were provided by Mohammad Mehdi Gilanian Sadeghi. The first draft of the manuscript was written by Reza Besharati and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

[1] R. Besharati, M. H. Rezvani, and M. M. G. Sadeghi, "An incentive-compatible offloading mechanism in fog-cloud environments using second-price sealed-bid auction," *Journal of Grid Computing*, vol. 19, no. 3, pp. 37–29, 2021.

[2] A. Lakhan, M. A. Mohammed, S. Kozlov, and J. J. Rodrigues, "Mobile-fog-cloud assisted deep reinforcement learning and blockchain-enable IoMT system for healthcare workflows," *Transactions on Emerging Telecommunications Technologies*, p. e4363, 2021.

[3] S. Shukla, M. F. Hassan, L. T. Jung, and A. Awang, "Architecture for latency reduction in healthcare internet-of-things using reinforcement learning and fuzzy based fog computing," in *Proceedings of the International Conference of Reliable Information And Communication Technology*, pp. 372–383, Springer, Johor, Malaysia, 2018, June.

[4] S. Chen, Y. Zheng, W. Lu, V. Varadarajan, and K. Wang, "Energy-optimal dynamic computation offloading for industrial IoT in fog computing," *IEEE Transactions on Green Communications and Networking*, vol. 4, no. 2, pp. 566–576, 2020.

[5] S. Chen, J. Chen, Y. Miao, Q. Wang, and C. Zhao, "Deep reinforcement learning-based cloud-edge collaborative mobile computation offloading in industrial networks," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 8, pp. 364–375, 2022.

[6] S. S. Lee and S. Lee, "Resource allocation for vehicular fog computing using reinforcement learning combined with heuristic information," *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 10450–10464, 2020.

[7] X. Zhang, Y. Xiao, Q. Li, and W. Saad, "Deep reinforcement learning for fog computing-based vehicular system with multi-operator support," in *Proceedings of the ICC 2020-2020 IEEE International Conference On Communications (ICC)*, pp. 1–6, IEEE, Dublin, Ireland, 2020, June.

[8] J. Shi, J. Du, J. Wang, J. Wang, and J. Yuan, "Priority-aware task offloading in vehicular fog computing based on deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 12, pp. 16067–16081, 2020.

[9] T. C. Hsu, H. Yang, Y. C. Chung, and C. H. Hsu, "A Creative IoT agriculture platform for cloud fog computing," *Sustainable Computing: Informatics and Systems*, vol. 28, Article ID 100285, 2020.

[10] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-efficient computation offloading and resource allocation in fog computing for internet of everything," *China Communications*, vol. 16, no. 3, pp. 32–41, 2019.

[11] F. Mashhadi, S. A. S. Monroy, A. Bozorgchenani, and D. Tarchi, "Optimal auction for delay and energy constrained

task offloading in mobile edge computing," *Computer Networks*, vol. 183, Article ID 107527, 2020.

[12] R. Besharati and M. H. Rezvani, "A prototype auction-based mechanism for computation offloading in fog-cloud environments," in *Proceedings of the 2019 5th Conference on Knowledge Based Engineering and Innovation (KBEI)*, pp. 542–547, IEEE, Tehran, Iran, February, 2019.

[13] J. Kim, T. Ha, W. Yoo, and J. M. Chung, "Task popularity-based energy minimized computation offloading for fog computing wireless networks," *IEEE Wireless Communications Letters*, vol. 8, no. 4, pp. 1200–1203, 2019.

[14] S. Misra and N. Saha, "Detour: dynamic task offloading in software-defined fog for IoT applications," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, 2019.

[15] V. Jafari and M. H. Rezvani, "Joint optimization of energy consumption and time delay in IoT-fog-cloud computing environments using NSGA-II Metaheuristic algorithm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1–24, 2021.

[16] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, Cambridge, MA, USA, 2018.

[17] T. Abbasi-khazaei and M. H. Rezvani, "Energy-aware and carbon-efficient vm placement optimization in cloud datacenters using evolutionary computing methods," *Soft Computing*, vol. 26, 2022.

[18] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous internet of things: model, applications and challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1722–1760, 2020.

[19] N. Wang and B. Varghese, "Context-aware distribution of fog applications using deep reinforcement learning," *Journal of Network and Computer Applications*, vol. 203, Article ID 103354, 2022.

[20] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Resource provisioning in fog computing through deep reinforcement learning," in *Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Bordeaux, France, May, 2021.

[21] P. Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach," *Future Generation Computer Systems*, vol. 110, pp. 1098–1115, 2020.

[22] G. M. S. Rahman, T. Dang, and M. Ahmed, "Deep reinforcement learning based computation offloading and resource allocation for low-latency fog radio access networks," *Intelligent and Converged Networks*, vol. 1, no. 3, pp. 243–257, 2020.

[23] F. Jazayeri, A. Shahidinejad, and M. Ghobaei-Arani, "Autonomous computation offloading and auto-scaling the in the mobile fog computing: a deep reinforcement learning-based approach," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 8, pp. 8265–8284, 2021.

[24] M. Chen, T. Wang, S. Zhang, and A. Liu, "Deep reinforcement learning for computation offloading in mobile edge computing environment," *Computer Communications*, vol. 175, pp. 1–12, 2021.

[25] J. Baek and G. Kaddoum, "Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks," *IEEE Internet of Things Journal*, vol. 8, no. 2, pp. 1041–1056, 2021.

[26] Y. Wei, F. R. Yu, M. Song, and Z. Han, "Joint optimization of caching, computing, and radio resources for fog-enabled IoT using natural actor–critic deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2061–2073, 2019.

[27] R. Zhu, S. Li, P. Wang, L. Li, A. Samuel, and Y. Zhao, "Deep reinforced energy efficient traffic grooming in fog-cloud elastic optical networks," in *Proceedings of the 2020 Optical Fiber Communications Conference And Exhibition (OFC)*, pp. 1–3, IEEE, San Diego, CA, USA, 2020, March.

[28] M. K. Pandit, R. N. Mir, and M. A. Chishti, "Adaptive task scheduling in IoT using reinforcement learning," *International Journal of Intelligent Computing and Cybernetics*, vol. 13, no. 3, pp. 261–282, 2020.

[29] C. K. Dehury and S. N. Srirama, "An efficient service dispersal mechanism for fog and cloud computing using deep reinforcement learning," in *Proceedings of the 2020 20th IEEE/ACM International Symposium On Cluster, Cloud And Internet Computing (CCGRID)*, pp. 589–598, IEEE, Melbourne, Australia, 2020, May.

[30] A. Mebrek, M. Esseghir, and L. Merghem-Boulahia, "Energy-efficient solution based on reinforcement learning approach in fog networks," in *Proceedings of the 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC)*, IEEE, Tangier, Morocco, 2019, June.

[31] K. Cui, B. Lin, W. Sun, and W. Sun, "Learning-based task offloading for marine fog-cloud computing networks of USV cluster," *Electronics*, vol. 8, no. 11, p. 1287, 2019.

[32] C. K. Dehury and S. N. Srirama, "Personalized service delivery using reinforcement learning in fog and cloud environment," in *Proceedings of the 21st International Conference On Information Integration And Web-Based Applications & Services*, pp. 522–529, Munich, Germany, 2019, December.

[33] G. Neelakantam, D. D. Onthoni, and P. K. Sahoo, "Reinforcement learning based passengers assistance system for crowded public transportation in fog enabled smart city," *Electronics*, vol. 9, no. 9, p. 1501, 2020.

[34] A. Nassar and Y. Yilmaz, "Reinforcement learning for adaptive resource allocation in fog RAN for IoT with heterogeneous latency requirements," *IEEE Access*, vol. 7, pp. 128014–128025, 2019.

[35] J. Jin and H. Li, "Real-time bidding with multi-agent reinforcement learning in display advertising," in *Proceedings of the 27th ACM International Conference WSDM 2017*, pp. 661–670, Cambridge, UK, February, 2017.

[36] W. Zhang, S. Yuan, and J. Wang, "Optimal Real-Time Bidding for Display Advertising," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1077–1086, New York, NY, USA, August, 2014.

[37] C. Han and R. Kan, "Real-time bidding by reinforcement learning in display advertising," in *Proceedings of the WSDM '17: Tenth ACM International Conference on Web Search and Data Mining*, pp. 661–670, ACM, Cambridge, UK, February, 2017.

[38] M. Du, "Improving real-time bidding using a constrained Markov decision process," in *Proceedings of the International Conference on Advanced Data Mining and Applications*, pp. 711–726, Springer, Chengdu China, October, 2017.

[39] S. Chen, R. Du, and S. Labi, "Scalable traffic signal controls using fog-cloud based multiagent reinforcement learning," 2021, https://arxiv.org/abs/2110.05564.

[40] A. A. Alli and M. M. Alam, "SecOFF-FCIoT: machine learning based secure offloading in Fog-Cloud of things for smart city applications," *Internet of Things*, vol. 7, Article ID 100070, 2019.

[41] A. Kishor, C. Chakraborty, and W. Jeberson, "Reinforcement learning for medical information processing over heterogeneous networks," *Multimedia Tools and Applications*, vol. 80, pp. 1–22, 2021.

[42] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 4951–4966, 2020.

[43] J. Y. Baek, G. Kaddoum, S. Garg, K. Kaur, and V. Gravel, "Managing fog networks using reinforcement learning based load balancing algorithm," in *Proceedings of the2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 1–7, IEEE, Marrakesh, Morocco, 2019, April.

[44] J. Santos, T. Wauters, B. Volckaert, and F. De Turck, "Reinforcement learning for service function chain allocation in fog computing," *Communication Networks and Service Management in the Era of Artificial Intelligence and Machine Learning*, pp. 147–173, John Wiley & Sons, Hoboken, NJ, USA, 2021.

[45] Y. Wang, X. Lin, and M. Pedram, "A nested two stage game-based optimization framework in mobile cloud computing system," in *Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, pp. 494–502, IEEE, San Francisco, CA, USA, March, 2013.

[46] M. Jia, J. Cao, and W. Liang, "Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks," *IEEE Transactions on Cloud Computing*, vol. 5, no. 4, pp. 725–737, 2017.

[47] J. Sztrik, *Basic Queueing Theory*, vol. 193, pp. 60–67, University of Debrecen, Faculty of Informatics, Debrecen, Hungary, 2012.

[48] G. Bolch, S. Greiner, H. De Meer, and K. S. Trivedi, *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*, John Wiley & Sons, Hoboken, NJ, USA, 2006.

[49] L. Liu, Z. Chang, T. Ristaniemi, and Z. Niu, "Multi-objective optimization for computation offloading in fog computing," *IEEE Internet of Things Journal*, vol. 5, 2017.

[50] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: a toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.

[51] M. H. Khoobkar, M. Dehghan Takht Fooladi, M. H. Rezvani, and M. M. Gilanian Sadeghi, "Partial offloading with stable equilibrium in fog-cloud environments using replicator dynamics of evolutionary game theory," *Cluster Computing*, vol. 25, pp. 1–33, 2022.

[52] M. H. Khoobkar, M. Dehghan Takht Fooladi, M. H. Rezvani, and M. M. Gilanian Sadeghi, "Joint optimization of delay and energy in partial offloading using dual-population replicator dynamics," *Expert Systems with Applications*, vol. 216, 2022.

[53] A. Babazadeh Nanehkaran and M. H. Rezvani, "An incentive-compatible routing protocol for delay-tolerant networks using second-price sealed-bid auction mechanism," *Wireless Personal Communications*, vol. 121, no. 3, pp. 1547–1576, 2021.

[54] S. Esfandiari and M. H. Rezvani, "An optimized content delivery approach based on demand–supply theory in disruption-tolerant networks," *Telecommunication Systems*, vol. 76, no. 2, pp. 265–289, 2021.

[55] V. Krishna, *Auction Theory*, Academic Press, Cambridge, MA, USA, 2009.