Hindawi
Security and Communication Networks
Volume 2021, Article ID 4348511, 9 pages
https://doi.org/10.1155/2021/4348511

Research Article

Randomization-Based Dynamic Programming Offloading Algorithm for Mobile Fog Computing

Wenle Bai,[1] Zhongjun Yang,[2] Jianhong Zhang,[3] and Rajiv Kumar[4]

[1]*Information Science and Technology, North China University of Technology, Shijingshan District, Beijing 100043, China*
[2]*School of Information Science and Technology, North China University of Technology, Shijingshan District, Beijing 100043, China*
[3]*North China University of Technology, Beijjing, China*
[4]*Department of Electronics and Communication As a Professor, Jaypee University of Information Technology, Waknaghat, India*

Correspondence should be addressed to Zhongjun Yang; 1076943446@qq.com

Received 22 July 2021; Accepted 10 August 2021; Published 31 August 2021

Academic Editor: Xin Liu

Offloading to fog servers makes it possible to process heavy computational load tasks in local devices. However, since the generation problem of offloading decisions is an N-P problem, it cannot be solved optimally or traditionally, especially in multitask offloading scenarios. Hence, this paper has proposed a randomization-based dynamic programming offloading algorithm, based on genetic optimization theory, to solve the offloading decision generation problem in mobile fog computing. The algorithm innovatively designs a dynamic programming table-filling approach, i.e., iteratively generates a set of randomized offloading decisions. If some in these sets improve the decisions in the DP table, then they will be merged into the table. The iterated DP table is also used to improve the set of decisions generated in the iteration to obtain the optimal offloading approximate solution. Extensive simulations show that the proposed DPOA can generate decisions within 3 ms and the benefit is especially significant when users are in multitask offloading scenarios.

1. Introduction

With the increasing popularity of smart devices, smart applications provide a rich user experience while placing stringent demands on computing power. Due to the limitations of mobile devices, it becomes a challenge to maintain the quality of service when traditional devices are faced with heavy computational demands. To solve this problem, mobile fog computing (MFC) [1] has been proposed to compensate for the lack of computing power in local devices. And, it alleviates the computing congestion of cloud by offloading tasks to fog nodes at the edge of the network, as well.

In MFC, there are two major problems to be solved. The first is how to determine whether a task should be offloaded or not. The other is how to balance the delay performance and energy consumption as much as possible, while offloading tasks. In order to solve the above problems, some offloading algorithms have been proposed in the literature. In [2], by modeling task generation as a Poisson process, the processing in the server is modeled as an $M/M/1$ model. A closed-form expression for the offloading delay is derived to find the global optimal solution with the lowest latency performance. Subdividing a task into the offloadable and non-offloadable part, the study in [3] constructs an opportunistic offloading model. A delayed offloading model is introduced in [4], in which tasks will wait as long as possible until the WIFI link is available to offload before the dead time arrives. Aiming at minimizing the energy consumption, the work in [5] develops an offloading interaction model based on the auction mechanism, while the authors of [6] propose an incentive propagation mechanism (IPM) algorithm.

In the traditional MFC network, a key research point is the joint design of the offloading decision and system resource allocation to optimize system effectiveness. The

study in [7] proposes a fair and energy-minimized task offloading (FEMTO) algorithm, which considers three important characteristics: the energy consumption of the offloading, the historical average energy of the F-AP, and the priority of the F-AP. The offloading decision problem and the uplink channel assignment problem are designed as a quadratic constrained quadratic programming (QCQP) problem in [8]. Then, the proposed matrix algorithm is used to solve the QCQP problem to obtain the suboptimal solution. The offloading problem in [9] is solved by transforming it into a nonconvex QCQP problem and then solving that with the proposed semidefinite relaxation algorithm.

With the boon of deep learning in recent years, some studies have combined deep learning with the MFC. The study in [10] proposes a joint computational offloading and resource allocation algorithm based on deep reinforcement learning (DRL), which is used for deriving a suboptimal solution for the optimization problem. A deep learning-based offloading algorithm is proposed in [11], which uses the trained DNN to generate offloading decisions for each task. The work in [12] introduces an empirical replay technique based on the reinforcement learning, which improves the convergence rate of the offloading algorithm. Similarly, the study in [13] adds a layer of the LSTM network to the deep Q network, which is used for predicting the amount of tasks to be received by the fog node at the next moment. Compared to the deep Q algorithm, the algorithm with the LSTM network can arrive at the optimal decision faster. Although these intelligent algorithms have very high offloading accuracy, the training of the network requires a long time. Meanwhile, these excessively complex algorithms will additionally increase the computational burden of IoT (Internet of Things) devices. Hence, they are not applicable for real-time computational offloading of MFC networks in time-varying environments.

In this work, attention is paid to the task offloading problem in an MFC network and a dynamic programming offloading algorithm (DPOA) based on randomization is proposed to solve the offloading decision generation problem. The achievements obtained in this paper are as follows:

(1) Randomization is introduced in the DPOA, where the system periodically generates a random set of offloading decisions of 1 or 0. And, these randomized decisions are used to construct a two-dimensional dynamic programming (DP) table, as well.

(2) The DP table is populated in a novel way, which avoids the double computation of the common decision cells in the generated random set. After enough iterations of the algorithm, the optimal approximate solution can be obtained from a query of the populated DP table.

(3) Extensive simulations show that the algorithm is highly responsive. And, the algorithm maintains optimal approximate decision generation within 3 ms for a wide range of system parameter settings.

The rest of this paper is as follows: the model of the MFC network and the closed-form expressions for latency and energy consumption are presented in Section 2, as well as the construction of an energy-latency weighted sum minimization problem. The DPOA is referred in Section 3. Section 4 mainly provides the analysis of simulations. And, the conclusions are given in the last section.

2. Network Model and Problem Formulation

2.1. Network Model. As shown in Figure 1, the network model investigates an MFC network consisting of multiple users, a single fog access point (F-AP), and a remote cloud server, where users interact with the F-AP through a wireless channel for data interaction. The F-AP and the cloud server are interconnected with an optical fiber. Assume that each user has N mutually independent tasks to compute, denoted as $N = \{1, 2, \ldots, N\}$. Each task is indivisible to the user and can be processed either locally, offloaded to a F-AP for processing or further to the cloud server.

The offloading decision of the n-th task at the local side is represented by the binary variable $X_F^n \in \{0, 1\}$. Then, $X_F^n = 0$ ($X_F^n = 1$) represents the task that will be processed at the local side (offloaded to the fog). Similarly, let $X_C^n \in \{0, 1\}$ represent the n-th task's decision that is on the fog side. $X_C^n = 0$ denotes the task that will remain on the fog side for computation, and, conversely, $X_C^n = 1$ represents the task being further offloaded to the cloud.

2.2. Problem Formulation. In the local computing mode, as the task is computed only in the local processor without involving data upload, the total energy and total time consumed are equal to those of the processor. The size of n-th task is denoted by M_n. Let κ and f_L represent the energy consumption per bit of task computed and the computation rate (cycle/s) at the local device, respectively. The energy consumption and latency of M_n computed locally are

$$E_L^n = \kappa M_n, \tag{1}$$

$$T_L^n = \frac{M_n \phi}{f_L}, \tag{2}$$

where ϕ indicates the number of CPU revolutions required that processes per bit of data (cycle/bit), which is determined by the CPU process. Without loss of generality, it is presumed that the all covered ϕ of the devices in this paper are all equal.

When $X_F^n = 1$ is in place, the task will be offloaded to the nearest F-AP over the wireless network. The task offloaded to the fog will be redistributed by the algorithm, and some will be further offloaded to the cloud server. According to [9, 11, 14, 15], it is known that the offloading backhaul data is extremely small, much smaller than of the offloaded, so the return cost is chosen to be ignored. Let the transmission rate between users and F-AP be B_F; then, the transmission time to the fog is

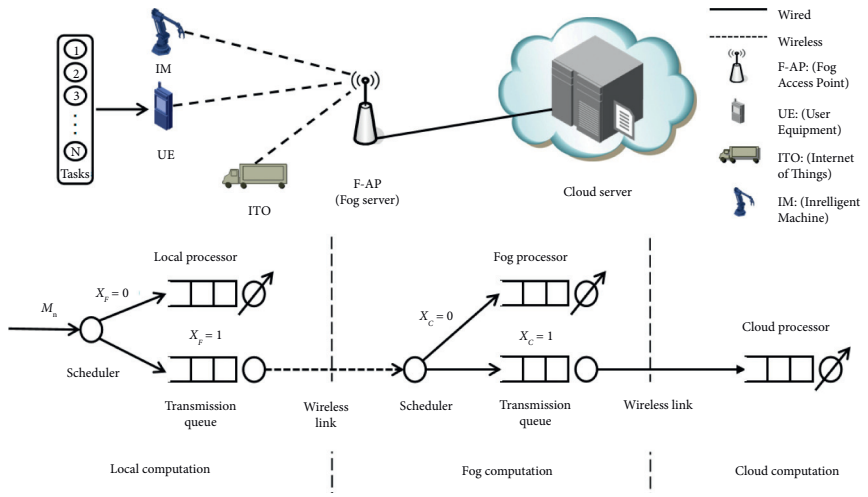

Figure 1: Network scenarios and offloading computation models for mobile fog computing.

$$T_{tF}^{n} = \frac{M_n}{B_F}. \tag{3}$$

With f_F denoting the processing rate of the fog server, the computation time consumed in the fog is

$$T_F^n = \frac{M_n \phi}{f_F}. \tag{4}$$

About the cloud computing model, F-APs are interconnected with the cloud server through the fiber and the transfer rate between both is expressed in terms of B_C. Then, the total time cost from the local side to the cloud can be recorded as

$$T_{tC}^{n} = \frac{M_n}{B_F} + \frac{M_n}{B_C}. \tag{5}$$

Representing the processing rate of the cloud server by f_C, the cloud computing time cost for the n-th task is

$$T_C^n = \frac{M_n \phi}{f_C}. \tag{6}$$

For energy consumption, as the essence of offloading is to handle complex or large tasks, which is difficult for the local side, this paper only considers the transmission and standby energy consumption in the local device while ignoring the energy cost at the server. Hence, the transmission energy consumption of the n-th task offloading to the fog is shown in

$$E_t^n = \alpha M_n, \tag{7}$$

where α is the energy consumption of the local device that transmits per bit of data.

In addition, the output of the local device is modeled as an $M/M/1$ queue and the system distributes the n+1th task only when the n-th has been processed. Meanwhile, if task n is offloaded, the local device will remain in the standby mode. Using P to present the standby power of the device, the standby energy consumption during offloading M_n can be expressed as E_w^n:

$$E_w^n = P \cdot \left[X_F^n \left(T_{tF}^n + T_F^n\right) + X_F^n X_C^n \left(T_{tC}^n + T_C^n\right)\right]. \tag{8}$$

For any user in the MFC network, the total latency and energy consumption of processing all tasks can be expressed, respectively, as follows:

$$T = \sum_{n=1}^{N} \left[\left(1 - X_F^n\right) T_L^n + X_F^n \left(1 - X_C^n\right)\left(T_{tF}^n + T_F^n\right) + X_F^n X_C^n \left(T_{tC}^n + T_C^n\right)\right], \tag{9}$$

$$E = \sum_{n=1}^{N} \left[X_F^n E_t^n + \left(1 - X_F^n\right) E_L^n + E_w^n\right]. \tag{10}$$

In order to minimize the total energy and time consumption to complete all tasks, this paper introduces a system utility Θ defined as a weighted sum of energy and latency, as shown in

$$\Theta(\mathbf{X}) = E + \gamma T, \tag{11}$$

where $\mathbf{X} = \{X_F^n, X_C^n \mid n \in \mathbf{N}\}$ and $\gamma \in [0, +\infty)$; (J/s) is the weight variable. When $\gamma = 0$, the system will focus only on the energy performance, and if a balance between energy and latency is to be maintained, then $\gamma = 1$ is required. For $\gamma > 1$, the system focuses more on the rapidity of the offloading.

After that, an optimization problem (P) is constructed to minimize the overall utility of the system $\Theta(\mathbf{X})$ with the help of optimizing the offloading decision $\mathbf{X}$ for each task. The optimization problem is as follows:

$$\text{P: } \Theta^*(\mathbf{X}) = \underset{\mathbf{X_F}, \mathbf{X_C}}{\text{minimize}} \left(E + \gamma T\right), \tag{12a}$$

$$\text{subject to } X_F \in \{0, 1\}, \tag{12b}$$

$$X_C \in \{0, 1\}, \tag{12c}$$

$$T \le T_{\text{limit}}, \tag{12d}$$

$$\text{gamma} \ge 0. \tag{12e}$$

where (12b) and (12c) constrain the binary nature of the offloading decision. Besides, (12d) is the offloading time limit. If the latency is greater than the time limit, then the decision will not be adopted. Moreover, the algorithm will add an extremely large penalty value, such as $\Theta^*(\mathbf{X}) = \Theta^*(\mathbf{X}) + \sigma$ where $\sigma = 10^9$, to the system utility at this point to force the decision to be eliminated in subsequent comparisons.

Generally speaking, a binary decision problem of this kind belongs to the N-P problem, which has no exact solution by default. In the next section, we will propose a dynamic programming offloading algorithm, which is based on randomized decision, to find the optimal approximate solution of such problems. Furthermore, the definitions of the symbols used in this paper are shown in Table 1.

3. DPOA

In this section, a dynamic programming offloading algorithm for the MFC network, the DPOA, is proposed, in which the two-dimensional DP table is constructed to pick the current optimal binary decision for each task.

Because of the similarity of the decision algorithms for the fog decision and the cloud decision, the algorithm is mainly used as an example at the fog server in this section. Since cloud-side decisions and fog-side decisions do not affect each other, when calculating fog-side decisions, we assume that all the decisions of the tasks in the cloud are 0, for example, $X_C^n \equiv 0$ $(\mathbf{X}_C = \{X_C^n \equiv 0 | n \in \mathbf{N}\})$.

To store and display the offloading decisions corresponding to each task, as shown in Figure 2, a DP table of size $N*N$ is constructed. Particularly, N is the total number of tasks. After the algorithm begins with H round-robin operations and a set of N randomly offloaded decision elements, denoted as $\Lambda_h, h \in \{1, 2, \ldots, H\}$ is generated in each cycle. In particular, there is no connection between the set of random decisions and the offloading task. While obtaining the generated set of offloading decisions for the current cycle, the system assigns the decisions to the previously constructed DP table according to a specific law. For example, 0 is assigned to the next horizontal cell while 1 is to the next vertical cell. Exceptionally, the starting position is (1, 2) if the first decision is 0 and (2, 1) is for 1. In addition, the construction of the DP table is based on the randomization theory in [9].

As an example, assuming $N = 6$, the algorithm constructs a $6 * 6$ DP table. Let all the decision sets generated by the first cycle be $\Lambda_1 = 010010$ and suppose the decision during the second cycle is $\Lambda_2 = 110101$ (as shown in Figure 2(a) with blue numbers) or $\Lambda_2 = 100110$ (in Figure 2(b) with red numbers) with two examples. As the first place of Λ_1 is 0, the starting cell is (1, 2). Since the second place of Λ_1 is 1, the next vertical cell of the previous cell (1, 2) will be assigned to 1, such as (2, 2) = 1. Similarly, with the third place being 0, the next horizontal cell of cell (2, 2) will be assigned to 0, i.e., (2, 3) = 0. The table is filled according to the above rules so as to get the table in Figure 2.

Once the decision set Λ_h is generated, it is first filled into the table according to the rules and the overall system utility $\Theta(\mathbf{X}_h)$ is calculated at the same time, in which $\mathbf{X}_F = \Lambda_h$ and $\mathbf{X}_C \equiv 0$. Obviously, if the number of iterations is large enough, there will be cases that the Λ_h has a common subset with the previous cells in the table with different decisions, as in (2, 2) in Figure 2(b). We will consider the utility values $\Theta(0, 0)$ and $\Theta(1, 0)$ for the cell under different offloading decisions, respectively, and select the decision corresponding to the lowest value to replace the original one. Then, the updated set is noted as Λ_h^*, followed by updating the overall system utility at this point as $\Theta(\mathbf{X}_h^*)$, specifically $\mathbf{X}_F = \Lambda_h^*$.

When all the elements in Λ_h are filled, the final $\Theta(\mathbf{X}_h^*)$ is compared with the system utility $\Theta(\mathbf{X}_{h-1}^*)$ of the h-1th cycle. In the special case, when h is equal to 1, then the $\Theta(\mathbf{X}_0)$ for the 0-th time is equal to the system utility corresponding to the full 0 set as Λ_0. If the utility value of the current loop is greater than of the previous one, the new set is replaced with the old like $\Lambda_h^* = \Lambda_{h-1}^*$. Furthermore, the utility value of the current loop is replaced with the old value as $\Theta(\mathbf{X}_h^*) = \Theta(\mathbf{X}_{h-1}^*)$, as well. Conversely, if the existing utility value is less than the value of the previous loop, the existing offloading decision is held without change.

Table 1: Symbols used in the paper.

Symbol	Meaning
M_n	Data size of users n-th task
X_F^n	Local-to-fog offloading decision of users n-th task
	$X_F^n = \begin{cases} 1 & \text{offloading} \\ 0 & \text{local} \end{cases}$
X_C^n	Fog-to-cloud offloading decision of users n-th task
	$X_C^n = \begin{cases} 1 & \text{cloud} \\ 0 & \text{fog} \end{cases}$
f_L	Local computing rate
f_F	Fog computing rate
f_C	Cloud computing rate
B_F	Bandwidth between the local and fog
B_C	Bandwidth between the fog and cloud
T_L^n	Local processing time of users n-th task
k	Energy cost of the device computed per bit of data
T_F^n	Fog processing time of users n-th task
α	Energy consumption per bit of data in local-to-fog transmission
T_C^n	Cloud processing time of users n-th task
P	Device standby power
T_{tF}^n	Transmission delay of n-th task from the local to fog
ϕ	Number of CPU revolutions required to process per bit of data
T_{tC}^n	Transmission delay of n-th task from the fog to cloud
T	Total delay of the user
E_L^n	Local energy consumption of users n-th task
γ	The relative importance of delay and energy
E_t^n	Uploaded energy cost of users n-th task
σ	System-set penalty value
E_w^n	Standby energy cost of the device when it offloads task n
E	Total energy cost of the user
H	Number of algorithm iterations
Λ_h	The set of random offloading decisions produced at the h-th iteration
Λ_h^*	The h-th offloading decision set after updating
η	Relative offloading accuracy rate

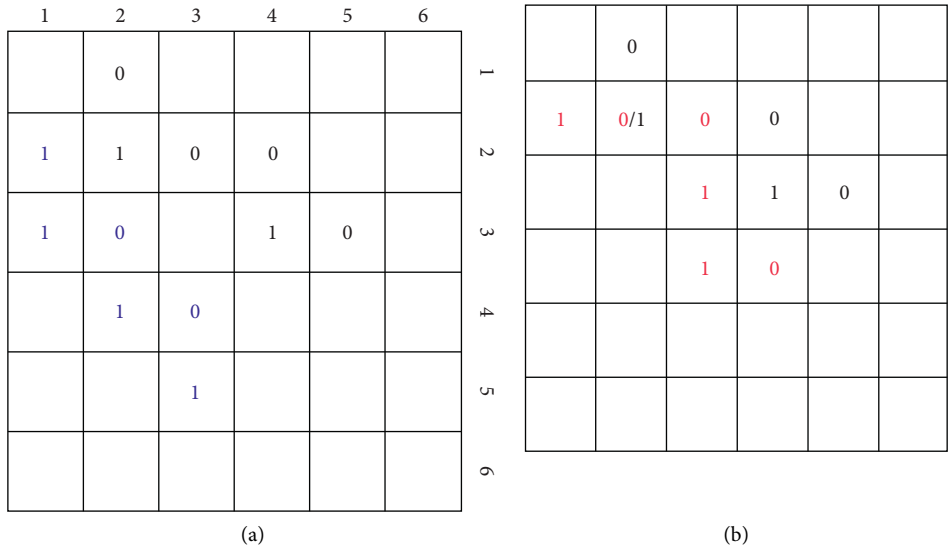

Figure 2: Two examples of filling out the DP table.

Having completed all the cycles, the decision set Λ_H^* after the H-th cycle is the optimal offloading decision $\mathbf{X}_F$ in the local-to-fog stage, i.e., $\mathbf{X}_F = \Lambda_H^*$. Obviously, if the number of cycles is sufficient, the constructed DP table will be filled up. Once the arbitrary set of offloading decisions generated at this point is filled into the table, the resulting updated set Λ_h^* is set as the corresponding optimal solution of the algorithm. And, the code of the DPOA is detailed in Algorithm 1.

For the decision algorithm on the cloud server, it is approximately the same as on the fog. Furthermore, if the fog decision for task n satisfies $X_F^n = 0$, then its cloud decision is also 0 ($\{X_C^n = 0 \mid X_F^n = 0, n \in \mathbf{N}\}$). In the next section, a large amount of simulations will be used to verify the speed and accuracy of the DPOA.

4. Evaluation and Simulation

In this section, extensive simulation is provided to evaluate the offloading performance of the DPOA. Specifically, it is assumed that the number of tasks to be offloaded by the user is 20, whose size is randomly distributed in the range of 2 MB to 20 MB. Furthermore, the computational energy consumption of the tasks is 3.25×10^{-7}J/bit [9] when being processed in the local device and 1.42×10^{-7}J/bit [9] is for the transmit energy per bit of the device. And, the standby power of the device is constant at 40 mw. Moreover, the processing speed of the local device, the fog server, and the cloud server is set to $f_L = 2.1 \times 10^8$cycle/s, $f_F = 5 \times 10^9$ cycle/s, and $f_C = 10 \times 10^9$cycle/s [9], respectively. The wireless transmission rate between the local and F-AP is 15 Mbps and from the fog to cloud is 40 Mbps [15]. Besides, $\phi = 100$cycle/s, $T_{\text{limit}} = 1200$ms, $H = 500$, and $\gamma = 1$J/s are further set.

The proposed DPOA is compared with the existing SRA (semidefinite relaxation approach) algorithm in the literature [9] and the Greedy algorithm, as shown in Figure 3. Besides, the data when the tasks are computed all locally and all in the cloud are added for reference. Figure 3 illustrates that offloading tasks can significantly reduce the processing latency and energy consumption compared to local computation. The addition of fog computation (DPOA, SRA, and Greedy) to the system can further reduce the utility compared to only cloud computation. From the data in Figure 3, it can be seen that the DPOA is able to find a better solution compared to the mentioned benchmark algorithm (SRA [9] and Greedy).

Here, we define the relative offloading accuracy rate of the DPOA, denoted by η:

$$\eta = \frac{\Theta(\text{SRA})}{\Theta(\text{DPOA})}. \tag{13}$$

where $\Theta(\text{SRA})$ denotes the system utility value corresponding to the decision generation by the SRA algorithm, while $\Theta(\text{DPOA})$ represents the value of the DPOA under the same parameters.

Figure 4 analyzes the performance of the algorithm at different numbers of iterations. In Figure 4(a), the relative accuracy rate increases gradually as the number of iterations continues to increase. When the number of H is higher than 500, the value of the rate tends to be smooth, i.e., the algorithm reaches the convergence state. As the number of iterations increases, the system utility value of the DPOA is decreasing, such as the line graph in Figure 4(b). And, when H is greater than 500, the system utility value tends to be stable. However, from the histogram in Figure 4(b), higher H values correspond to higher decision time points for the system, so that a single increase in the number of iterations may weaken the rapidity of the algorithm.

In Figure 5, we consider the offloading when the user is in a wireless transmission rate variation scenario with a transmission rate B_F range of 5 Mbps to 25 Mbps. As can be seen in Figure 5(a), the energy under the DPOA is much lower than that of the full cloud computing and outperforms that of the benchmark algorithm. In a poor communication scenario, such as $B_F = 5$Mbps, all-local computation is more cost-effective than offloading in terms of energy consumption because of the relatively heavy energy

Input: number of tasks N; size of tasks $M_n, n \in \mathbf{N}$;
Output: optimal offloading decision set Λ_H^*;
(1) **Initialization:** initialize the DP table and set the T_{limit};
(2) **for** $h = 1, 2, \ldots, H$ **do**
(3) Generate random offloading decision sets Λ_h;
(4) Fill in the DP table according to the rules and calculate the system utility $\Theta(\mathbf{X}_h)$ for Λ_h;
(5) **if** *(the filled decisions have a common set and not the same as the cells in the previous table)* **then**
(6) Calculate the system utility corresponding to this decision and the original decision, respectively;
(7) Select the decision corresponding to the lower value to replace the original decision;
(8) Renew Λ_h^* and $\Theta(\mathbf{X}_h^*)$;
(9) **else**
(10) Execute the following: $\Lambda_h^* = \Lambda_h$ and $\Theta(\mathbf{X}_h^*) = \Theta(\mathbf{X}_h)$;
(11) **if** $(\Theta(\mathbf{X}_h^*) > \Theta(\mathbf{X}_{h-1}^*))$ **or** $(T > T_{\text{limit}})$ **then**
(12) Execute the following: $\Lambda_h^* = \Lambda_{h-1}^*$ and $\Theta(\mathbf{X}_h^*) = \Theta(\mathbf{X}_{h-1}^*)$;

Algorithm 1: DPOA.

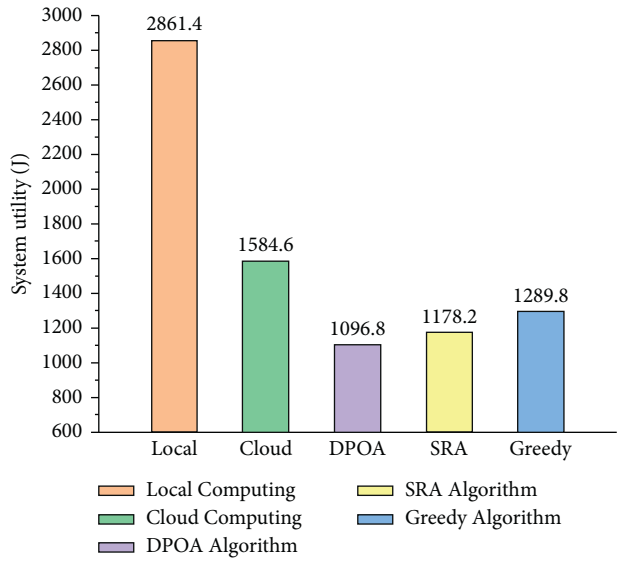

Figure 3: Comparison of the system utility for different algorithms.

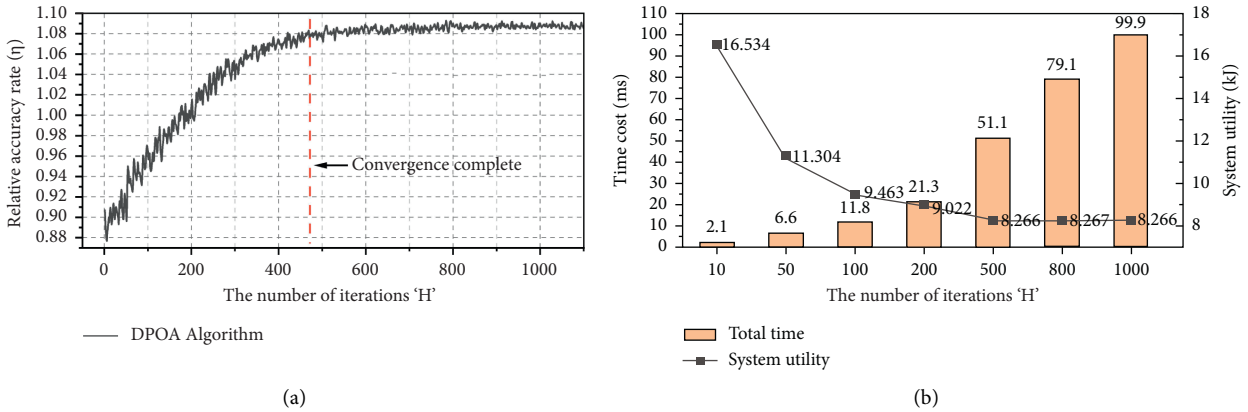

Figure 4: Performance under different number of iterations "H": (a) convergence of the DPOA; (b) total time cost and system utility.

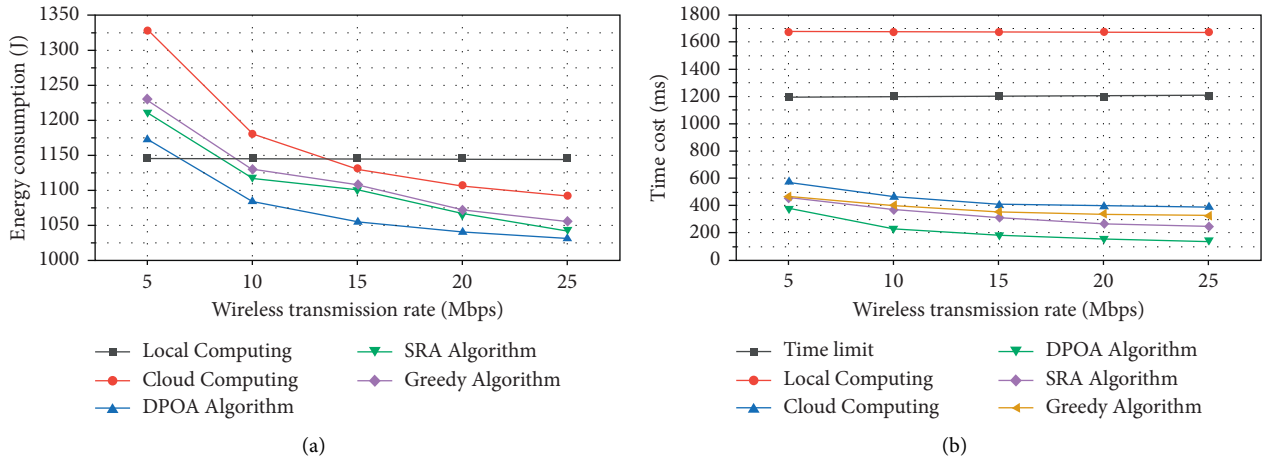

Figure 5: Comparison of the DPOA and the benchmark algorithms (SRA [5] and Greedy), when time limit = 1200: (a) energy consumption; (b) time cost.

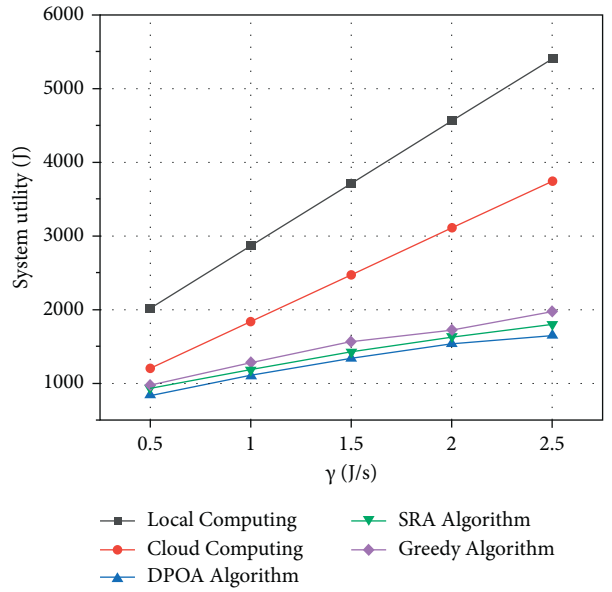

Figure 6: System utility values of different offloading algorithms under different γ.

consumption required to offload. However, it is noticed in Figure 5(b) that the all-local computation exceeds the time limit, so the optimal solution is the DPOA in comparison.

The relationship between the time cost and the wireless transmission rate is exhibited in Figure 5(b). As the transmission rate increases, the advantage of offloading computation over non-offloading becomes gradually obvious and the latency of the system to perform offloading computation decreases.

As illustrated in Figure 6, the performance of the algorithm for different γ has been investigated. It can be seen from the figure that as the value of γ increases, the system utility also increases gradually. However, the utility of the DPOA is always lower than that of the SRA and the Greedy, which means that the DPOA gives a superior approximate solution in comparison. Based on Figures 3, 5, and 6, it can be concluded that the DPOA consistently outperforms the mentioned benchmark algorithm in terms of comprehensive offloading performance.

More specifically, the time cost of this algorithm in generating offloading decisions under different numbers of tasks is also analyzed, as shown in Figure 7. Not surprisingly, it can be noticed that the decision-elapsed time of the DPOA increases almost linearly as the number of tasks increases from 10 to 35. At the same time, the average decision time curve shows that with the changing number of tasks, the algorithm can steadily maintain the optimal offloading decision within 3 ms. The above experiments illustrate that the algorithm can handle the massive task offloading problem without reducing its own efficiency and is suitable for multitask offloading scenarios.

In Figure 8, we further investigate the impact of the time limit (T_{limit}) on the energy efficiency of the algorithm. Adopting the energy consumption in the case of all-local

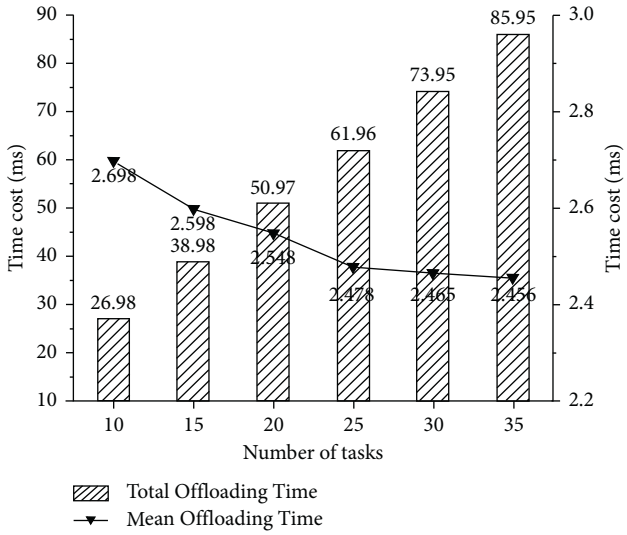

Figure 7: Time cost for generating decision under different number of tasks using the DPOA.

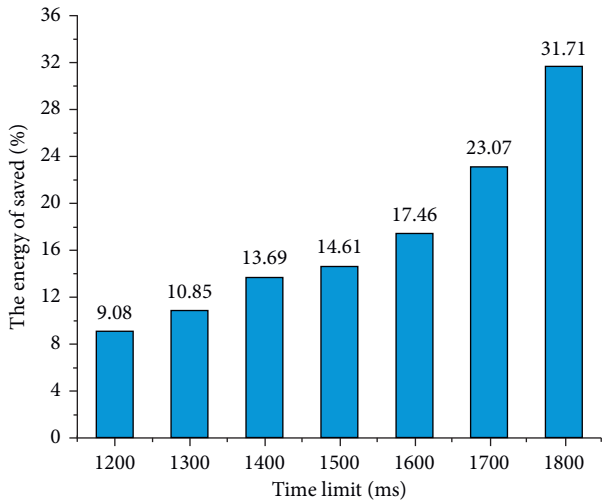

Figure 8: Percentage of energy saved by using the DPOA compared to the all-local computing method, under different time limits.

computing as the benchmark, Figure 8 shows the percentage energy savings of T_{limit} relative to the benchmark. It can be seen from the figure that as the time limit is gradually relaxed, more tasks can be offloaded to the server and thus more energy can be saved. Compared with the all-local computing model, the DPOA can reduce the energy consumption of the local device processing tasks by about 9% to 32% while meeting the time constraints.

5. Conclusion

Aiming at the offloading decision generation problem in MFC, this paper proposes a randomization-based dynamic programming offloading algorithm, DPOA, to solve this problem. The algorithm uses randomization to cyclically generate a random set of offloading decisions while constructing an overall system utility $\Theta(\mathbf{X})$, including the total energy and time cost, to dynamically populate the DP table by minimizing the system utility. The simulation results verify that the proposed algorithm is accurate and outperforms the existing benchmark algorithm. Furthermore, the DPOA can better adapt to scenarios with changing transmission rates. However, at higher network rates, it will offload as many tasks as possible to the server, thus converging quickly to the optimal solution. Besides, the algorithm can generate near-optimal offloading decisions in 3 ms and its time cost does not increase drastically with the number of tasks. In conclusion, we hope that this DPOA can be applied in decision time-sensitive MFC scenarios, such as smart IoT in 6G, to improve the efficiency of real-time system offloading.

Data Availability

The underlying data supporting the results of this study can be found at the official website of Beijing Natural Science Foundation.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by Beijing Natural Science Foundation, Haidian Original Innovation Joint Fund Project (No. L182039).

References

[1] P. Habibi, M. Farhoudi, S. Kazemian, S. Khorsandi, and A. Leon-Garcia, "Fog computing: a comprehensive architectural survey," *IEEE Access*, vol. 8, Article ID 69105, 2020.

[2] M. Xu, Z. Zhao, M. Peng, Z. Ding, T. Q. S. Quek, and W. Bai, "Performance analysis of computation offloading in fog-radio access networks," in *Proceedings of the 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, Shanghai, China, May 2019.

[3] J. Wang, T. Lv, P. Huang, and P. T. Mathiopoulos, "Mobility-aware partial computation offloading in vehicular networks: a deep reinforcement learning based scheme," *China Communications*, vol. 17, no. 10, pp. 31–49, Oct. 2020.

[4] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Transactions on Mobile Computing*, vol. 17, no. 2, pp. 461–474, 1 Feb. 2018.

[5] 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, Tehran, Iran, March 2019.

[6] L. Yang, H. Zhu, H. Wang, H. Qian, and Y. Yang, "Incentive propagation mechanism of computation offloading in fog-enabled D2D networks," in *Proceedings of the 2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–4, Shanghai, China, November 2018.

[7] G. Zhang, F. Shen, Z. Liu, Y. Yang, K. Wang, and M. Zhou, "FEMTO: fair and energy-minimized task offloading for fog-

enabled IoT networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, June 2019.

[8] M. Mukherjee, S. Kumar, Q. Zhang et al., "Task data offloading and resource allocation in fog computing with multi-task delay guarantee," *IEEE Access*, vol. 7, Article ID 152911, 2019.

[9] M. Chen, B. Liang, and M. Dong, "A semidefinite relaxation approach to mobile cloud offloading with computing access point," in *Proceedings of the 2015 IEEE 16th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pp. 186–190, Stockholm, Sweden, July 2015.

[10] 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, Dec. 2020.

[11] X. Zhu, S. Chen, S. Chen, and G. Yang, "Energy and delay co-aware computation offloading with deep learning in fog computing networks," in *Proceedings of the 2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pp. 1–6, London, UK, October 2019.

[12] X. Wang, X. Wei, and L. Wang, "A deep learning based energy-efficient computational offloading method in Internet of vehicles," *China Communications*, vol. 16, no. 3, pp. 81–91, March 2019.

[13] F. Jiang, R. Ma, C. Sun, and Z. Gu, "Dueling deep Q-network learning based computing offloading scheme for F-ran," in *Proceedings of the 2020 IEEE 31st Annual International Symposium on Personal*, pp. 1–6, London, UK, September 2020.

[14] M. A. Sharkh and M. Kalil, "A dynamic algorithm for fog computing data processing decision optimization," in *Proceedings of the 2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–6, Dublin, Ireland, June 2020.

[15] W. Bai, Z. Ma, Y. Han et al., "Joint optimization of computation offloading, data compression, energy harvesting, and application scenarios in fog computing," *IEEE Access*, vol. 9, Article ID 45462, 2021.