Multi-task offloading scheme for UAV-Enabled Fog Computing networks

Xujie LI (lixujie@hhu.edu.cn)
Hohai University https://orcid.org/0000-0001-5486-5702
Lingjie Zhou
Hohai University
Ying Sun
Hohai University

Research

Keywords: UAV, Fog computing networks, Task offloading, Firework algorithm

Posted Date: March 16th, 2020

DOI: https://doi.org/10.21203/rs.3.rs-17296/v1

Version of Record: A version of this preprint was published on November 4th, 2020. See the published version at https://doi.org/10.1186/s13638-020-01825-y.

Multi-task offloading scheme for UAV-Enabled Fog Computing Networks

Xujie Li[1,2,3]*, Lingjie Zhou[1], Ying Sun[1]

[1]*College of Computer and Information Engineering, Hohai University, Nanjing, 210098, China*
[2]*National Mobile Communications Research Laboratory, Southeast University, Nanjing, 210096, china*
[3]*Key Lab for Wireless Sensor network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, 200050, China*
* Correspondence: lixujie@hhu.edu.cn

Abstract: In UAV-enabled fog Computing networks, how to efficiently offload multiple tasks to the computing nodes is a challenge combinatorial optimization problem. In this paper, in order to optimize the total delay for the UVA-Enabled Fog Computing networks, a simple scheduling algorithm and a multi-task offloading scheme based on fireworks algorithm (FWA) are proposed. First, the system model of multiple tasks offloading in UVA-Enabled fog computing networks is described in detail. Then, a simple scheduling algorithm is proposed to optimize the delay of the tasks allocated to a single node. Based on the scheduling algorithm, a multi-task offloading scheme for all tasks and all computing nodes is presented. Finally, simulation results show that the performance of proposed scheduling algorithm and offloading strategy outperform than that of genetic algorithm and random algorithm. This result can provide an effective optimization for multi-task offloading in UVA-Enabled Fog Computing networks.

Keywords: UAV, Fog computing networks, Task offloading, Firework algorithm

1. Introduction

1.1. Motivation

In recent years, with the popularization of smart phones and various new applications, wireless data traffic has increased thousands of times [1,2]. The increasing demand of users has promoted the continuous progress of communication technology. Meanwhile， Fifth Generation (5G) has gradually started commercialized [3]. Since the future wireless communication system is no longer only refer to the simple ground communication, but wants to make full use of the multi-dimensional information of space, ground and sea to realize the comprehensive management of complex networks of time and space. In the backgrounds, UAV communications have received extensive attention worldwide [4,5].

With the advent of ultra-reliable low-latency communications, delay-sensitive and reliability-aware task demands are also rapidly increasing. Computation-intensive tasks or applications, such as image or video-based applications, require higher processing power and energy consumption resources [6]. For UAV networks, due to their own computing resources and battery energy constraints, intensive tasks will affect the real time operation of the UAV system, or even cause tasks to be blocked. Therefore, task offloading is an effective way to solve this problem for UAV networks and has attracted the attention and research of many scholars.

1.2. Related Research

It is well known that UAVs can utilize their own mobility to get rid of space constraints and establish the flexible communication, but their limited computing resources and battery power also make UAVs endure a challenge. In [7], an offloading algorithm is proposed to assist UAV in performing computationally intensive tasks. This algorithm provides two methods for task offloading. The first offloading method is airborne offloading, where UAV can offload their computing tasks to UAV nearby with computing and energy resources available. The second offloading method is ground offloading, which allows tasks to be offloaded from a multi-level edge

cloud unit connected with a ground station to an edge cloud server. In [8], the authors focus on a scenario of a group of small UAVs performing an exploration mission. They give a comprehensive proof of the existence of Nash equilibrium and present a distributed algorithm that converges to this equilibrium.

In paper [9], combining UAV-aided communication and MEC is presented as a promising paradigm, and it can cope with the surging demands for Big Data processing in UAV-aided IoT applications. The energy consumption for accomplishing the tasks can be effectively reduced based on the proposed algorithm. In paper [10], the authors propose a novel UAV-enabled MEC system which can interact with IoT devices, UAV and edge clouds (ECs). To improve the Quality of Service, optimization problem which minimize the weighted sum of the service delay of all IoT devices and UAV energy consumption is formulated and solved. In paper [11], the authors present a new architecture for UAV clustering to enable efficient multi-modal multi-task offloading to overcome the heavy overhead of real-time interaction. Then the computing, caching, and communication resources are collaboratively optimized with AI-based decision making.

In paper [12], the authors study the task offloading problem between the IMDs and the UAV to minimize the overall energy consumption for UAV-aided edge computing networks. The task offloading decision-making, bit allocation during transmission, and the UAV trajectory are jointly optimized to solve the problem. In paper [13], the authors optimize the deployment of UAVs in the consideration of their number and locations. Meanwhile, task scheduling is also optimized to provide high quality services for all mobile users. For each mobile user, the problem that its task is executed locally or on a UAV is discussed and analyzed in detail. Then a two-layer optimization method to presented to solve the problem. In paper [14], the authors solve the problem of offloading heavy computation tasks of UAVs while achieving the best possible tradeoff between energy consumption, time delay, and computation cost. And the scenario of a fleet of small UAVs performing an exploration mission is considered. The problem is formulated as a non-cooperative theoretical game with N players and three pure strategies. In paper [15], an energy-efficient computation offloading with an emphasis on physical-layer security is presented and discussed. Several energy-efficiency problems for secure UAV-MEC systems are formulated and transformed to convex problems. Then their optimal solutions are obtained. In paper [16], the authors propose a novel game-theoretic and reinforcement learning framework for computational offloading for the mobile edge computing network which are operated by multiple different service providers. Then the network operation is modeled to a two-level hierarchical model. The upper one is formulated as a cooperative game and the lower one is modeled as several noncooperative subgames.

1.3. Contributions

In the work mentioned above, the conventional optimization methods are mostly used to solve the task-offloading problem for MEC or Fog networks. And few works consider the scenario of multiple tasks. In our scenario, multiple tasks can be allocated to multiple computing nodes and one computing node may need to be allocated multiple tasks. Therefore, how to allocate these tasks to multiply computing nodes and how to schedule these tasks allocated one computing node are crucial problems to be solved. In this paper, an efficient schedule algorithm for the tasks to allocated one computing node and a multi-task offloading scheme based on improving fireworks algorithm for UAV-Enabled fog computing networks are proposed. The main contributions of our work are as follows.

1) We propose an efficient schedule algorithm for the tasks to allocated one computing node.

2) A multi-task offloading scheme based on improving fireworks algorithm for UAV-Enabled Fog Computing networks.

3) Simulation results validate the efficiency of the proposed scheme.

1.4. Paper Organization

The remainder of this paper is organized as follows. In Section 2, system model of multiple tasks offloading for UAV-Enabled fog computing networks is presented, which includes the

air-to-ground (A2G) channel model and the task offloading model. Then the method we used in this paper is introduced and the problem is formulated in Section 3. In Section 4, a task scheduling algorithm and a multi-task offloading scheme based on firework algorithm is proposed to improve the performance of the system. Simulation results are provided and discussed in Section 5 and the paper is concluded in Section 6.

2. system model

In this paper, one UAV and multiple computing nodes coexist in the UAV networks as shown in Figure 1. We assume that N computing nodes (CNs) are uniformly distributed in the cellular with the radius of R. One UAV with M different tasks is hovering in the center of the cellular. The UAV offloads M tasks to the CNs, and the CNs return the results to the UAV after completing the calculation. Of course, only the CNs that their Quality of Service (Qos) can be satisfied have the chance to be selected to compute the tasks. We denote ith CN as CN_i $i \in \{1,2,3,\cdots,N\}$ and jth task as TASK_j $j \in \{1,2,3,\cdots,M\}$, respectively. In our scenario, we consider that the number of tasks is greater than the number of CPs. So one CN is usually allocated to multiple tasks. It is assumed that the Orthogonal Frequency Division Multiplexing (OFDM) channels are adopted for the communication between the UAV and the CNs in the networks. In Figure1, the blue block and the yellow block represent the communication capabilities and computing capabilities of the CNs, respectively. Therefore, how to efficiently offload these tasks to CNs is a challenge problem.

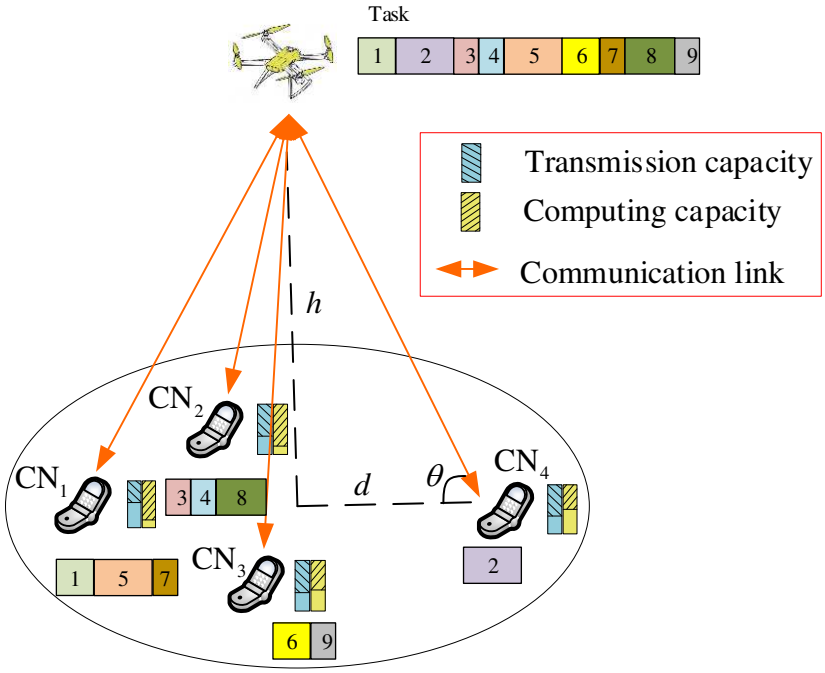

Figure 1. System model

2.1. Channel Model

For A2G communication, it is well known that a classical UAV communication mode can be adopted. And the path loss model can be represented as [17]:

$$\mathrm{PL} = \frac{A}{1 + a\exp\left[-b(\arctan\left(\frac{h}{d}\right) - a)\right]} + 10\log\left(h^2 + d^2\right) + B \tag{1}$$

Where $A = \eta_{\mathrm{Los}} - \eta_{\mathrm{NLos}}$, $B = 20\log f + 20\log\left(4\pi/c\right) + \eta_{\mathrm{NLos}}$, h is the height of the UAV, d denotes the distance between CN and the UAV's projection on the ground, a and b are constants related to the propagation environment, which are usually suburban, urban, dense urban and highrise urban. $\eta_{\mathrm{Los}}, \eta_{\mathrm{NLos}}$ are the average additional loss of the free propagation space loss which changes with the environment, c is the propagation speed of light and f is the carrier frequency.

In general, we can assume that the allocated transmission power at the UAV for every CN is same to P_t. Then the signal-to-noise ratio of CN_i can be written as

$$SNR_i = \frac{P_t / PL_i}{N_0} \tag{2}$$

Here, PL_i is the path loss between the UAV and CN_i, N_0 is the noise power. Next, the data transmission rate of CN_i can be calculated as

$$R_i = B \cdot \log_2\left(1 + SNR_i\right) \tag{3}$$

Here, B is the sub-channel bandwidth. Then, we denote the data size of each task as $\mathrm{C} = \left\{C_1, C_2, ...C_j, ..., C_N\right\}$.

To taking into full account the diversity of the tasks, three different distributions of data size of the tasks are considered in our paper: uniform distribution, Gaussian distribution, and Pareto distribution [18]. Meanwhile, the transmission delay $m_{i,j}$ for offloading TASK_j to CN_i is mainly determined by the data size of the task and the transmission rate R_i. Then $m_{i,j}$ can be expressed as

$$m_{i,j} = \frac{C_j}{R_i} \tag{4}$$

2.2. Computing model

The computing ability of the CN_i is only related to the frequency f_i of it's CPU. We denote the required CPU cycles to process each bit of data as η_i. If the CN_i computes TASK_j, then the computing delay $p_{i,j}$ can be obtained as

$$p_{i,j} = \frac{C_j \eta_i}{f_i} \tag{5}$$

3. Method and problem formulation

Multi-task offloading is a crucial issue for UAV-enabled fog computing networks. Therefore, how to allocate these tasks to multiply computing nodes and how to schedule these tasks allocated one computing node are crucial problems to be solved. For the first problem, it is a combinatorial optimization problem, we propose algorithm based on the fireworks algorithm. For the second problem, an efficient schedule algorithm is presented to solve it.

In UAV-enabled fog computing networks, UAV may encounter multiple computing tasks requirement. And the sizes of these tasks are different in general. Meanwhile, for the CN, their communication and computing abilities are also unequal. For the task offloading scheme, delay is a very important performance index. The delay consists of three parts: the transmission delay, computing delay and the delay of result return. And the delay of the result return to the UAV can be

ignored because the data size of the return result of the task is usually very small [19]. Next, we define a binary decision variable $\mu_{i,j}$ to indicate if $TASK_j$ is allocated to CN_i. We have

$$\mu_{i,j} = \begin{cases} 1 & TASK_j \text{ is allocated to } \mathrm{CN}_i \\ 0 & TASK_j \text{ is not allocated to } \mathrm{CN}_i \end{cases} \tag{6}$$

Considering that the CN usually has a certain storage space, the allocated tasks to the CN can be offloaded continuously. That is, UAV can immediately offload next task after last task is offloaded. But for the computing, the CP must wait until the transmission of the task is completed. Therefore, there may be a gap between the transmission of next task and computing of current task. Now, we denote the number of tasks allocated to CN_i as M_i. And we have $\sum_{i=1}^{N} M_i = M$. Then how to calculate the delay of completing these tasks for CN_i is a very challenge problem. And how to schedule these tasks for CN_i is also a crucial problem to be solved.

We denote the delay that CN_i complete these allocated tasks as T_i. Then system delay can be written as an optimization problem

$$T = \min \max_{i \in N} \{T_i\} \tag{7}$$

$$\text{s.t.} \quad \sum_{i=1}^{N} \mu_{i,j} = 1 \quad j \in \{1, 2, 3, \cdots, M\}$$

4. Multi-task offloading scheme for UAV-enabled fog computing networks

4.1. Task scheduling algorithm

As mentioned above, task scheduling in the CNs has a great impact on system performance in term of the delay. Actually, it is not easy to get the expression of T_i. Considering the relationship between the communication ability and the computing ability of the CN, the scenario can be divided into two possibility: communication delay is smaller than computing delay or the computing delay is smaller than communication delay.

Next, we propose a simple scheduling algorithm. If communication delay is smaller than computing delay, we sort the tasks from smallest to biggest. If the computing delay is smaller than communication delay, we sort the tasks from biggest to smallest. Next, we analyze the mechanism in detail.

1. Communication delay is smaller than computing delay ($m_{i,j} < p_{i,j}$) and sort the tasks from smallest to biggest.

Firstly, we consider very friendly condition. We assume that the communication delay of every task is smaller than the computing time of the previous task, that is $m_{j+1} > p_j$. In this case, the total delay of CN_i is the transmission time of all tasks allocated to CN_i plus the computing time of the last task, as shown in Figure 2 (a). For the example in Figure 2 (a), we can obtain the delay as

$$T_{total} = \sum_{j=1}^{3} m_j + p_3 \tag{8}$$

But in other conditions, such as $TASK_2$ has not completely arrived when the computing of $TASK_1$ is finished, as shown in Figure 2(b). In this case, CN must wait until the communication of next task is completed. And there is a waiting time $w_j = m_j - p_{j-1}$ before next task will be computed. Then the delay can be written as $T_{total} = m_1 + m_2 + p_3 + p_4$ for this example.

2. Computing delay is smaller than communication delay ($p_{i,j} < m_{i,j}$) and sort the tasks from biggest to smallest

Similarly, we consider very friendly condition firstly as shown in Figure 2(c). In this situation, we also assume that the computing time of the task is greater than the transmission time of its previous task, that is $p_j > m_{j+1}$. Then the total delay is the computing time of all tasks allocated to CN_i and transmission time of the first task. So, it can be written as

$$T_{total} = \sum_{j=1}^{3} p_j + m_1 \tag{9}$$

But in other conditions, such as $TASK_2$ has not completely arrived when the computing of $TASK_1$ is finished, as shown in Figure 2(d). In this case, CN must wait until the communication of next task is completed. And there is a waiting time $w_j = m_j - p_{j-1}$ before next task will be computed. Then the delay can be also written as $T_{total} = m_1 + m_2 + p_3 + p_4$ for this example.

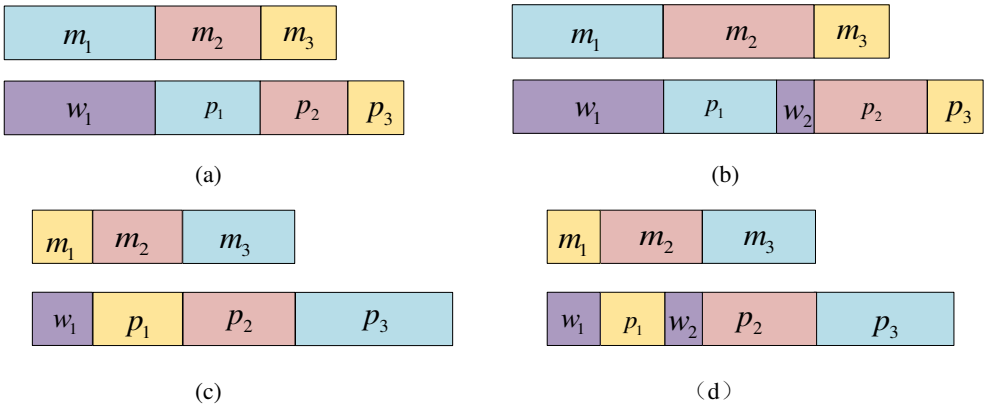

Figure 2. The diagram of delay under different scenarios

Based on the analysis above, the proposed scheduling algorithm is as follows.

Algorithm 1 Scheduling algorithm for the tasks allocated to one CN

Step1：Initialization parameters (task number *L*, task size *C*, transmission ability of the CN, computing ability of the CN)

Step2: Calculate the transmission delay $m_{i,j}$ and computing delay $p_{i,j}$ for the tasks allocated to the CN.

Step3: compare $m_{i,j}$ and $p_{i,j}$. If $m_{i,j} > p_{i,j}$, sort the tasks set M_i from smallest to biggest, else sort the tasks set M_i from biggest to smallest

Step3: Set the counters for the delay.

$$T_{M\_total} = m_1$$

$$T_{P\_total} = m_1$$

Step4： Compare transmission delay m_i 和 computing delay p_i：

for (*i*=1 to *L*-1)

$$T_{M\_total} = T_{M\_total} + m_{i+1}$$

$$T_{P\_total} = T_{P\_total} + p_i$$

if $T_{M\_total} > T_{P\_total}$

$T_{P\_total} = T_{M\_total}$

end

end

Step5: $T_{total} = T_{P\_total} + p_L$

Output: the total delay T_i of the CN_i

4.2. The preliminary of fireworks algorithm

Fireworks Algorithm is a relatively novel swarm intelligence algorithm proposed by Professor Tan Ying and others inspired by the explosion of fireworks in the night sky [20]. The firework algorithm constructs a fitness function according to the optimization objective function of the problem, and each solution corresponds to a firework position. Randomly initialize a certain number of locations to set off fireworks. Then we can simulate the explosion of fireworks so generate explosive sparks scattered near the fireworks. Next randomly select a small number of fireworks to implement Gaussian explosion operation. Then the position of the explosion spark corresponds to the new generated solution. Thus, the optimal solution of the problem will be obtained through multiple iterations. The process of fireworks explosion is also the process of optimizing in the feasible solution space along with the operations such as explosion sparks and mutated sparks.

4.3. Multi-task offloading scheme based on fireworks algorithm

In this subsection, a multi-task offloading scheme based on fireworks algorithm is proposed. The scheme can search optimal solutions more thoroughly due to its explosion mechanism. Compared with other classical intelligent algorithms, it has a very good performance. The proposed scheme is as follows:

1. Initialization coding
   In this paper, the feasible solution is mapped to real number coding for the firework algorithm. Initialization coding corresponds to the initial firework position. The specific coding strategy is as follows. Assume that a single UAV carries M tasks with different amount of computations. Then the position of firework is encoded as an M-dimensional vector, denoted as $X_i = (x_1, ..., x_m, ..., x_M)$, $x_m \in [1, N]$. And each element represents a task. For example, the vector $X_i = (1,4,2,2,1,3,1,2,3)$ represents a feasible solution that is randomly generated for the case that 9 tasks need to be offloaded to 4 computing nodes. The position sequence number in the row vector represents the index of all tasks. For X_i, the corresponding task offloading strategy is that $TASK_1$, $TASK_5$, and $TASK_7$ are offloaded to CN_1, $TASK_3$, $TASK_4$, $TASK_8$ are offloaded to CN_2, $TASK_6$, $TASK_9$ are offloaded to CN_3, and $TASK_2$ is offloaded to CN_4, as shown in the system model of Figure 1. In our algorithm, we denote the number of fireworks as N_c.
2. Explosion operator
   We denote explosion radius as A_i and number of explosion sparks as S_i, respectively. And the sparks are generated around the sparks are generated around. Compared with bad fireworks, fireworks with good fitness values have more explosion sparks and a smaller explosion radius [21]. A_i and S_i can be determined by the fitness value of each firework in the population according to the following formula:

$$A_i = A \cdot \frac{f(X_i) - y_{\min} + \varepsilon}{\sum_{i=1}^{N} (f(X_i) - y_{\min}) + \varepsilon} \tag{10}$$

$$S_i = S \cdot \frac{y_{\max} - f(X_i) + \varepsilon}{\sum_{i=1}^{N} (y_{\max} - f(X_i)) + \varepsilon} \tag{11}$$

Where, $f(X)$ is the fitness function of the feasible solution X, $y_{\min} = \min\left(f\left(X_i\right)\right)$ and $y_{\max} = \max\left(f\left(X_i\right)\right)$ are the minimum and maximum fitness values in the current firework population, respectively. A and S are constants used to adjust the explosion radius and the number of explosions, respectively. ε is a smoothing parameter used to avoid the case that the denominator is equal to 0.

Meanwhile, the fireworks with good fitness values often generate excessive sparks so as to fall into the local optimal values, and the fireworks with poor fitness values do not generate sparks again. To avoid these cases, we limit the number of sparks S_i as follows [21]:

$$S_i = \begin{cases} \text{round}(a * M), S_i < aM \\ \text{round}(b * M), S_i > bM \\ \text{round}(S_i), \text{otherwise} \end{cases} \tag{12}$$

Where, a and b are constants which are used to specify the upper and lower limits of the number of sparks generated, $\text{round}(\cdot)$ is rounding functions.

3. Mutation operator

In order to maintain the diversity of the population, a mutation operator is introduced to generate new sparks. We denote Gaussian spark number as *GN*. Then *GN* fireworks will be selected from the N_c fireworks to implement Gaussian mutation. For each selected firework, we randomly select kth element, and perform the following operations for each selected element to form the new sparks:

$$X_i(k) = \text{round}\left(X_i(k) \cdot e\right) \tag{13}$$

Where, $e \sim N(1,1)$ is a Gaussian variance with the mean of 1 and the variance of 1 [21].

4. Correction

The new sparks generated in the above two operations may escape the feasible region. So for every new generated spark, it should be checked. If the kth element of spark X_i is out of the feasible region, the correction value of X_i can be obtained by the mapping rule of the following formula:

$$X_i(k) = \text{round}\left(x_L + \left|X_i(k)\right| \% \left(x_U - x_L\right)\right) \tag{14}$$

Where, x_U and x_L are the upper and lower boundary values of the elements, respectively.

5. Elite strategy

In order to keep the optimal solutions to the next generation, at the end of each iteration, the original firework population and the above-mentioned generated sparks by explosion and mutation operations are retained in a temporal population. Next, we sort these sparks according

to their fitness functions in descending order. Then the elite selection strategy is adopted to select top N_c sparks as the next generation of population.

Base on the analysis above, a multi-task offloading scheme based on fireworks algorithm for UAV-Enabled fog computing networks is presented as follows.

Algorithm 2 multi-task offloading scheme based on fireworks algorithm

Step1: Initialization parameters (Radius of the cell *R*, task number *M*, CN number *N*).

Initialize positions $X_i \ \ i \in \{1,2,3,\cdots,N\}$ of Nc fireworks.

Step2: According to the objective function values obtained in the previous step, the number and radius of the explosion are calculated by formula (10) and (11), and the explosive sparks are generated.

Step3: *GN* fireworks were selected for gaussian variation operation according to formula (13) to generate gaussian sparks.

Step4: According to formula (14), the out-of-bounds sparks generated in the above two steps are modified.

Step5: For every spark, execute Algorithm 1 to calculate the delay for each CN_i, then calculate the optimal solution according to formula (7).

Step6: The elite strategy is adopted to select top N_c fireworks and enter the next iteration.

Step7 : If the number of iterations is reached, the algorithm is ended and the optimal solution is obtained. If not, return Step2.

---

5. Results and Discussion

In this section, a MATLAB simulation was conducted to evaluate the performance of the proposed scheme. We assume that *N* computing nodes (CNs) are uniformly distributed in the cellular with the radius of R. one UAV with *M* different tasks is hovering in the center of the cellular. The simulation parameters are shown in Table1.

Table 1. Simulation parameters

parameter	value	parameter	value
Radius of the cell *R*	500m	CPU cycles of CP η	2000cycle/bit
Number of CP *N*	3	CPU frequency of CN	$[2,4]\times10^7$ cycle/s
Number of TASK *M*	8	Number of fireworks N_c	5
Transmitting power of UAV P_t	10dBm	Blast number adjusting factor *S*	30
Noise power N_0	-105dBm	Blast radius adjusting *A*	8
Bandwidth of channel *B*	0.2MHz	Gaussian spark number *GN*	5
Altitude of UAV *h*	360m		

Figure 3 and Figure 4 describe the delay of the CN in two scenarios that communication delay is smaller than computing delay and the computing delay is smaller than communication delay

under different task scheduling algorithms. In order to verify the advantages of the proposed scheduling algorithm, the random algorithm and the exhaustive algorithm are compared with it in this simulation. Dark blue, light blue, yellow, and brown respectively represent random algorithm, small to large algorithm, large to small algorithm, and exhaustive algorithm. It can be seen from the figure that no matter the size of tasks obeys the gaussian distribution, uniform distribution or Pareto distribution, when the node's computing delay is smaller than communication delay, the delay of the small to large algorithm is close to exhaustion, that is, close to the optimal strategy, which is obviously better than the large to small algorithm and random algorithm. Conversely, when the node's computing delay is smaller than communication delay, the total delay of the large to small algorithm is close to exhaustion algorithm. Therefore, no matter what distribution the size of tasks follows, our proposed scheduling algorithm can get very good performance.

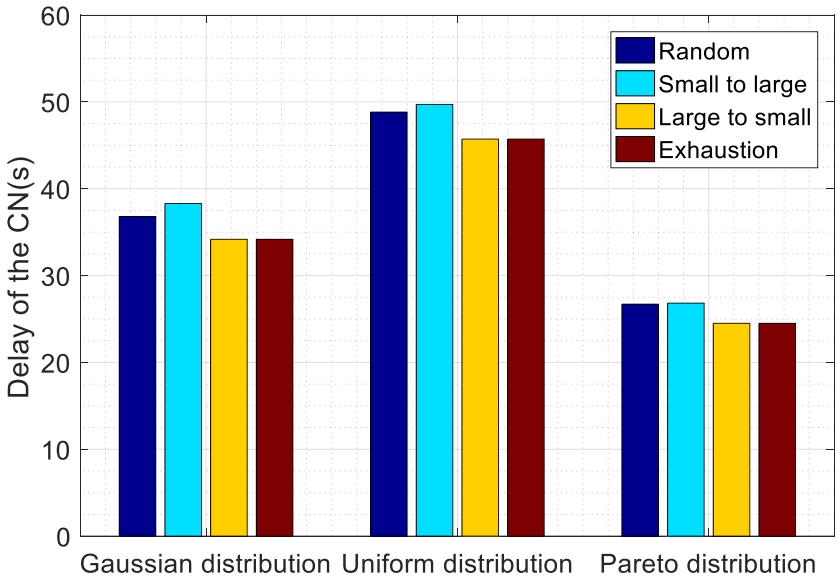

Figure 3. Delay of CN in the scenario communication delay is smaller than computing delay

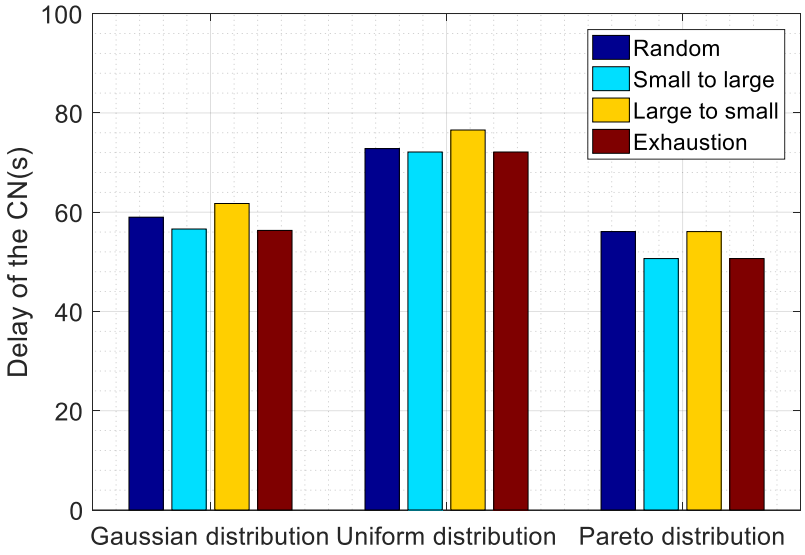

Figure 4. Delay of CN in the scenario computing delay is smaller than communication delay

Figure 5 and Figure 6 show the CDF of the delay of single node under different scheduling algorithms when the size of task follows a gaussian distribution in the above two scenarios. The results are obtained by Monte Carlo simulations. When the computing delay of the node is smaller than the communication delay or the communication delay of the node is smaller than the computing delay, our proposed scheduling algorithm is close to the exhaustive algorithm. This further validate the efficiency of our proposed scheduling algorithm.

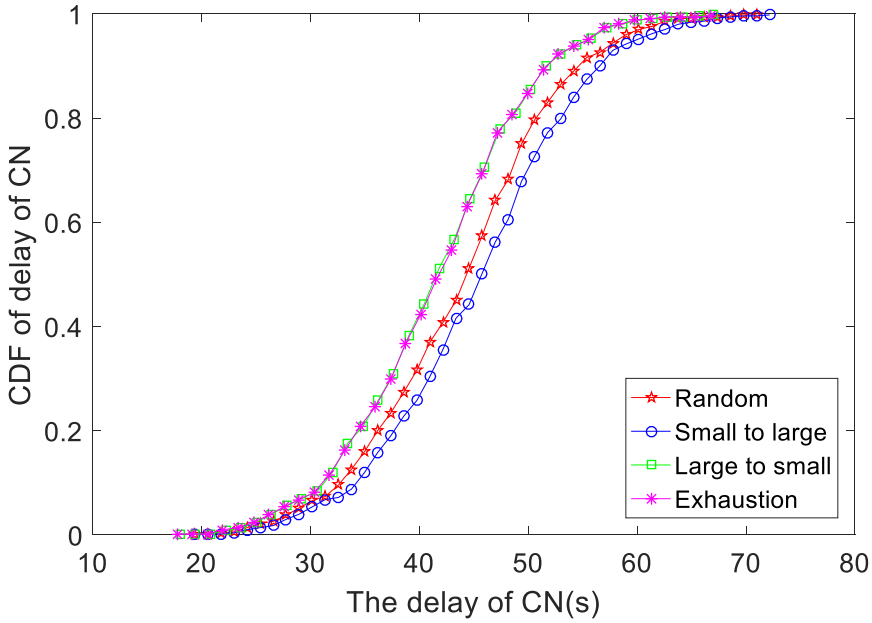

Figure 5. CDF of delay of CN in the scenario communication delay is smaller than computing delay

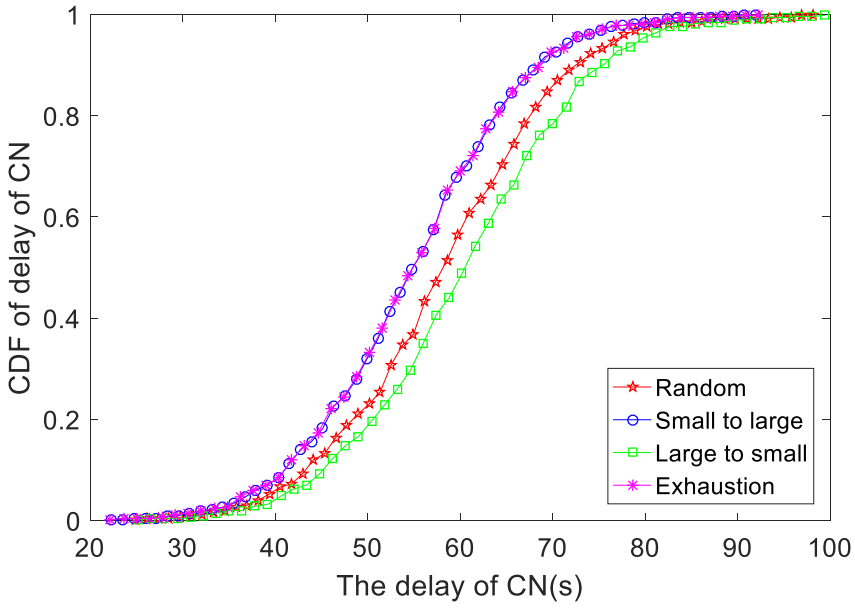

Figure 6. CDF of delay of CN in the scenario computing delay is smaller than communication delay

Next, the total system delay is simulated and discussed. The genetic algorithm (GA) and the random algorithm (RA) are also considered as the benchmark. From Figure 7, it can be intuitively seen that the performance of the RA is worst. Compared with the RA, GA can get better performance. But for GA, it is difficult to jump out of the local optimal solution sometimes. But for our proposed algorithm based on the schedule algorithm and firework algorithm, it can get best

performance compared with other two algorithm. Meanwhile, the proposed algorithm can get fast convergence.

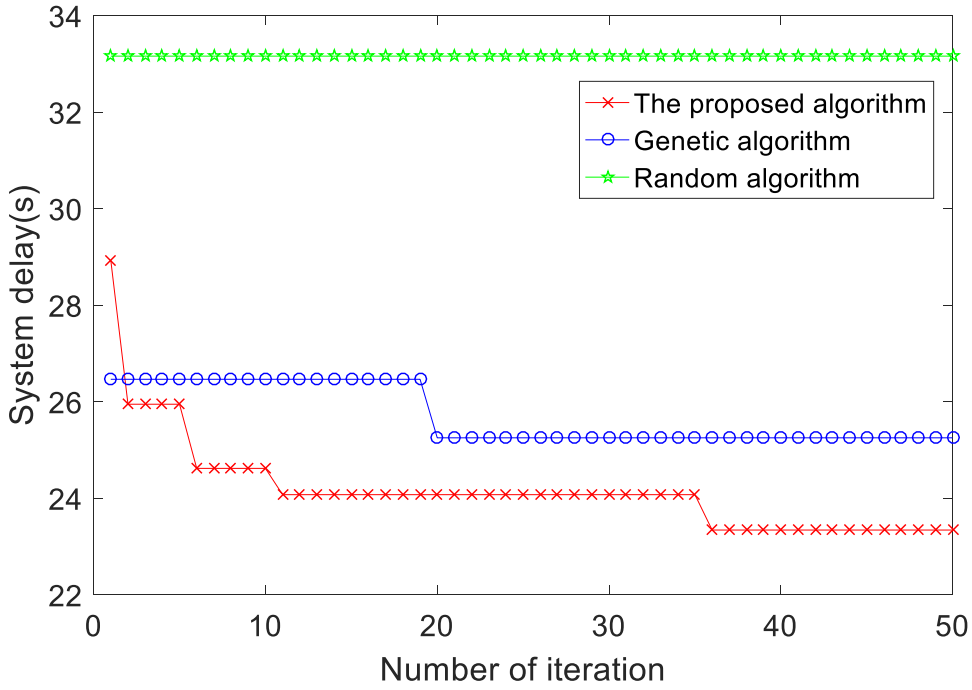

Figure 7. The total system delay under different algorithms

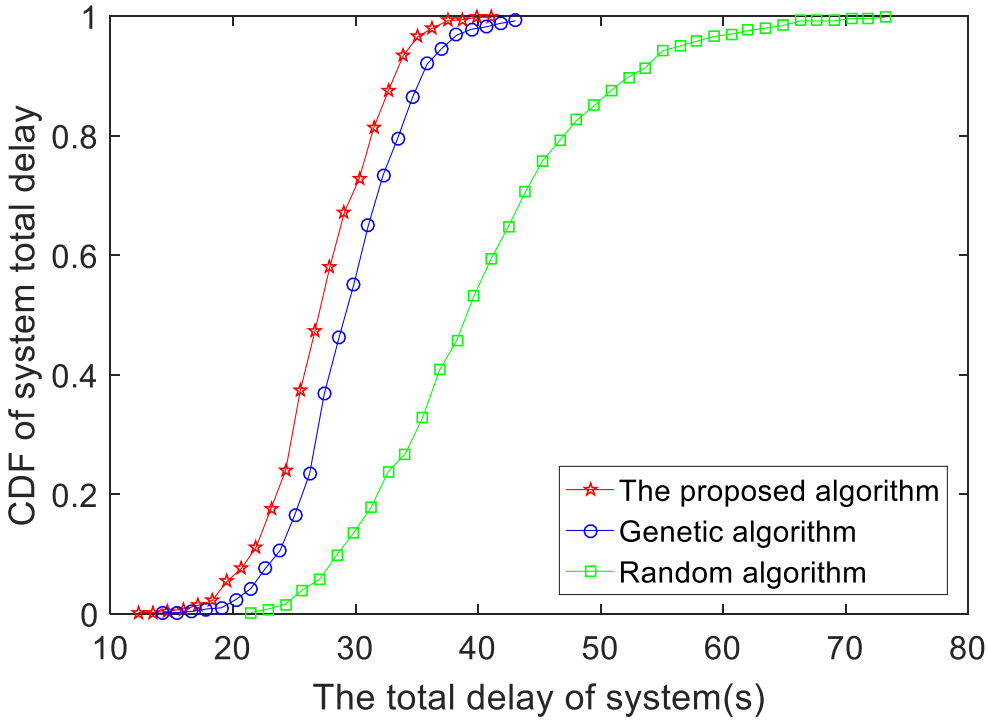

Figure 8. CDF of the total system delay under different algorithms

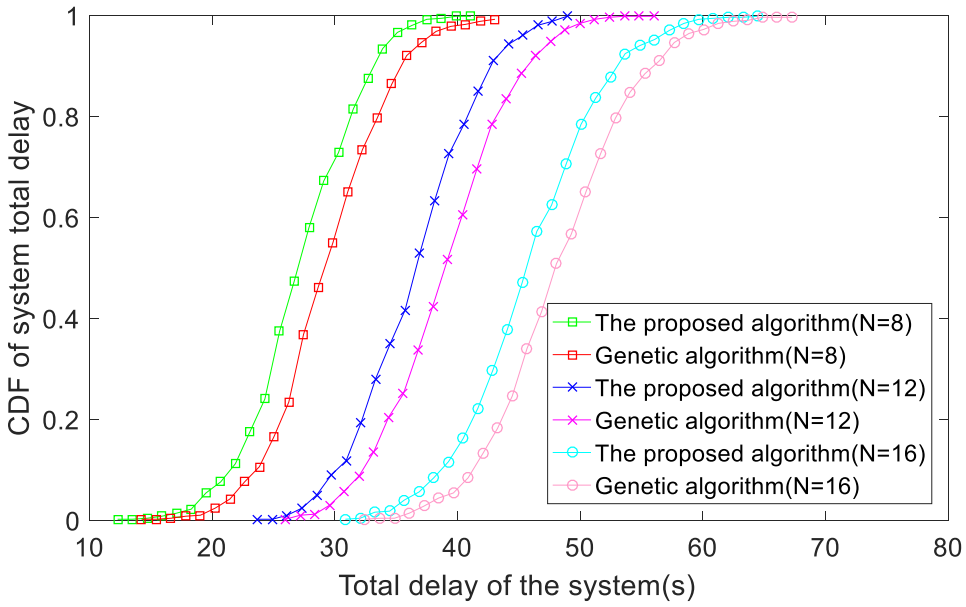

Figure 9. CDF curve of system delay with different number of tasks

Figure 8 shows the CDF curves of the total system delay under different algorithms. From this figure, it can be seen that the performance of the RA is worst due to the randomness of allocation. And the total system delays of these three algorithms are about in range of 11s to 41s. Compared with the RA and GA, the proposed algorithm can reduce the total system delay by 15s and 2s, respectively. This shows the efficiency of our proposed algorithm.

Figure 9 shows the relationship between the number of tasks and the total system delay of the GA and our proposed algorithm. It can be observed from the figure that the system delay increases with the increasing of the number of tasks for the two algorithms. Meanwhile, the system delay will increase by about 10s for each additional 4 tasks. Compared with the GA, our proposed offloading algorithm is always better than the GA. This show the advantages of our proposed algorithm for the multi-task offloading problem.

6. Conclusions

Aiming at the multi-task offloading problem in the UAV-enabled fog Computing networks, this paper proposes a scheduling algorithm and a multi-task offloading scheme. Firstly, multiple tasks offloading for UAV-enabled fog computing networks is modeled, and the problem is formulated. Then a scheduling algorithm is proposed to optimize the delay for one CN. Considering the novelty and advantage of fireworks algorithm, it is introduced. Based on the scheduling algorithm and fireworks algorithm, a multi-task offloading scheme based on improving fireworks algorithm for UAV-Enabled Fog Computing networks is proposed. It can get good performance in term of system delay. Finally, simulation results validate the efficiency of the proposed scheme.

Abbreviations

FWA: fireworks algorithm; 5G: Fifth Generation; ECs: edge clouds; CN: computing node; A2G air-to-ground; Qos: Quality of Service; OFDM: Orthogonal Frequency Division Multiplexing; GA: genetic algorithm; RA: random algorithm;

Funding

This work was supported in part by the open research fund of National Mobile Communications Research Laboratory Southeast University(No. 2020D16), in part by the Provincial Key Research and Development Program of Jiangsu under Grant BE2019017, in part by the Six Talent Peaks project in Jiangsu under Grant DZXX-008, and in part by the Open Research Fund Key Laboratory of Wireless Sensor Network and Communication, Chinese Academy of Sciences, under Grant 20190914.

Authors' contributions

All three authors have contributed to the work presented in this paper. The approach was conceived by Xujie Li. Lingjie Zhou proposed the algorithm. Ying Sun analyzed the data. All authors worked collaboratively on writing the main text.

Competing interests

The authors declare that they have no competing interests.

Author details
[1]College of Computer and Information Engineering, Hohai University, Nanjing, 210098, China
[2]National Mobile Communications Research Laboratory, Southeast University, Nanjing, 210096, china
[3]Key Lab for Wireless Sensor network and Communication, Shanghai Institute of Microsystem and Information Technology, Chinese Academy of Sciences, Shanghai, 200050, China

References

1. X. Li, Y. Sun, L. Zhou, A. Qi and S. Zhou, A Resource Allocation Scheme Based on Immune Algorithm for D2D-Based Vehicular Communication Networks. IEEE Access. **7**, 122536-122543(2019)
2. X. Li, Y. Sun, L. Zhou, Y. Xu, S. Zhou, A resource allocation scheme based on predatory search algorithm for ultra-dense D2D communications. Wireless Networks. **6**, 1-9(2019)
3. H. Shakhatreh et al., Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges. IEEE Access. **7**, 48572–48634(2019)
4. J. Ye, C. Zhang, H. Lei, G. Pan and Z. Ding, Secure UAV-to-UAV Systems With Spatially Random UAVs," IEEE Wireless Commun. Lett. **8**(2), 564-567(2019)
5. Y. Zeng, Q. Wu and R. Zhang, Accessing From the Sky: A Tutorial on UAV Communications for 5G and Beyond, IEEE, **107**(12), 2327-2375(2019)
6. M. Mukherjee et al., Task Data Offloading and Resource Allocation in Fog Computing With Multi-Task Delay Guarantee. IEEE Access. **7**, 152911-152918(2019)
7. A. A. Ashraf Ateya, A. Muthanna, R. Kirichek, M. Hammoudeh and A. Koucheryavy, Energy- and Latency-Aware Hybrid Offloading Algorithm for UAVs. IEEE Access. **7**, 37587-37600(2019)
8. M. Messous, S. Senouci, H. Sedjelmaci and S. Cherkaoui, A Game Theory Based Efficient Computation Offloading in an UAV Network. IEEE Trans. Veh. Technol. **68**(5), 4964-4974(2019)
9. H. Guo and J. Liu, UAV-Enhanced Intelligent Offloading for Internet of Things at the Edge. IEEE Trans. Ind. Inform. **16** (4), 2737-2746(2020)
10. Z. Yu, Y. Gong, S. Gong and Y. Guo, Joint Task Offloading and Resource Allocation in UAV-Enabled Mobile Edge Computing. IEEE Internet Things. Early Access, 1-1 (2020)
11. L. Hu, Y. Tian, J. Yang, T. Taleb, L. Xiang and Y. Hao, Ready Player One: UAV-Clustering-Based Multi-Task Offloading for Vehicular VR/AR Gaming. IEEE Network. 33(3), 42-48(2019)
12. J. Xiong, H. Guo and J. Liu, Task Offloading in UAV-Aided Edge Computing: Bit Allocation and Trajectory Optimization. IEEE Commun. Lett. **23**(3), 538-541(2019)
13. Y. Wang, Z. Ru, K. Wang and P. Huang, Joint Deployment and Task Scheduling Optimization for Large-Scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing. IEEE Trans. Cybernetics. Early Access, 1-1 (2019)
14. M. Messous, S. Senouci, H. Sedjelmaci and S. Cherkaoui, A Game Theory Based Efficient Computation Offloading in an UAV Network. IEEE Trans. Veh. Technol. **68**(5), 4964-4974(2019)
15. T. Bai, J. Wang, Y. Ren and L. Hanzo, Energy-Efficient Computation Offloading for Secure UAV-Edge-Computing Systems. IEEE Trans. Veh. Technol. **68**(6), 6074-6087(2019)
16. A. Asheralieva and D. Niyato, Hierarchical Game-Theoretic and Reinforcement Learning Framework for Computational Offloading in UAV-Enabled Mobile Edge Computing Networks With Multiple Service Providers. IEEE Internet Things. **6**(5), pp. 8753-8769(2019)
17. A. Hourani, K. Sithamparanathan, and S. Lardner, Optimal LAP altitude for maximum coverage. IEEE

Wireless Commun. Lett. **3**(6), 569572(2014)

18. M. Chen and Y. Hao, Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. IEEE J. Sel. Area Commun. **36**(3), 587-597(2018)
19. J. Liu and Q. Zhang, Code-partitioning offloading schemes in mobile edge computing for augmented reality. IEEE Access, **7**, 11222–11236(2019)
20. Tan, Ying, Yu, Chao, Zheng and Shaoqiu, Introduction to Fireworks Algorithm. International Journal of Swarm Intelligence Research, **4**(4), 39-70(2013)
21. Y. Tan and Y Zhu, Fireworks Algorithm for Optimization. Paper presented at Advances in Swarm Intelligence, First International Conference (ICSI), Beijing, China, 12-15, June 2010

Figures

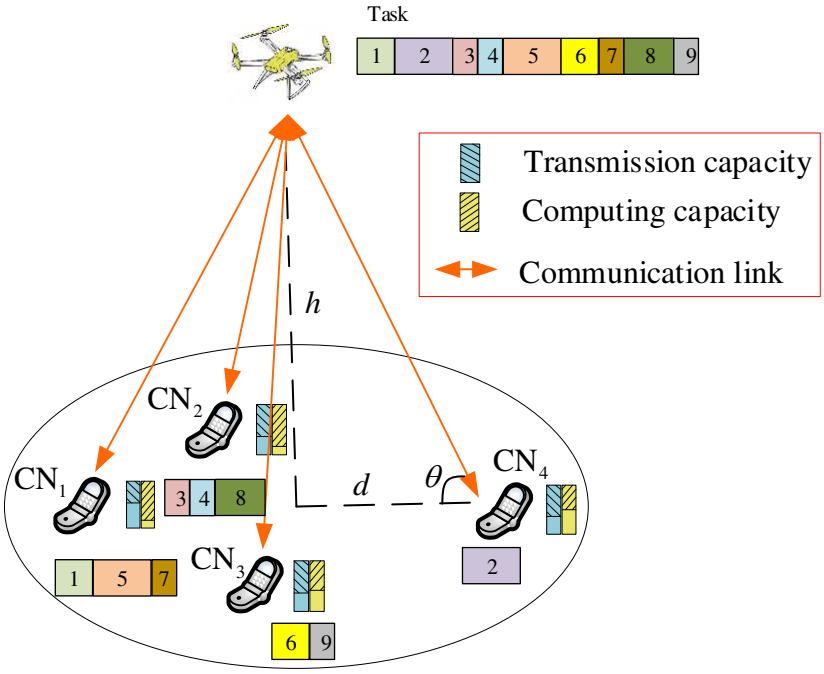

Figure 1. System model. One UAV and multiple computing nodes coexist in the UAV network. We assume that N computing nodes (CNs) are uniformly distributed in the cellular with the radius of *R*. One UAV with M different tasks is hovering in the center of the cellular. The UAV offloads M tasks to the CNs, and the CNs return the results to the UAV after completing the calculation.

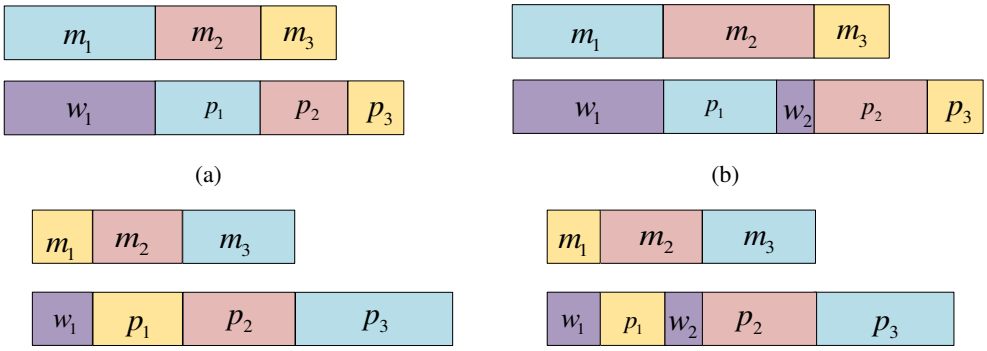

Figure 2. The diagram of delay under different scenarios. We denote the communication delay and computing delay as m_i and p_i, respectively. If communication delay is smaller than computing delay, we sort the tasks from smallest to biggest. If the computing delay is smaller than communication delay, we sort the tasks from biggest to smallest.

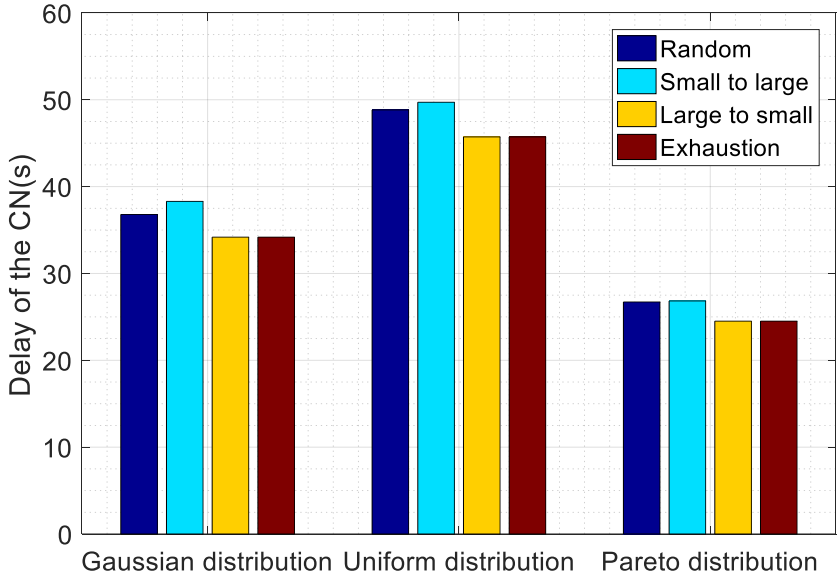

Figure 3. Delay of CN in the scenario communication delay is smaller than computing delay. This figure describes the delay of the CN that communication delay is smaller than computing delay under different task scheduling algorithms. It can be seen from the figure that no matter the size of tasks obeys the gaussian distribution, uniform distribution or Pareto distribution, when the node's computing delay is smaller than communication delay, the delay of the small to large algorithm is close to the optimal strategy.

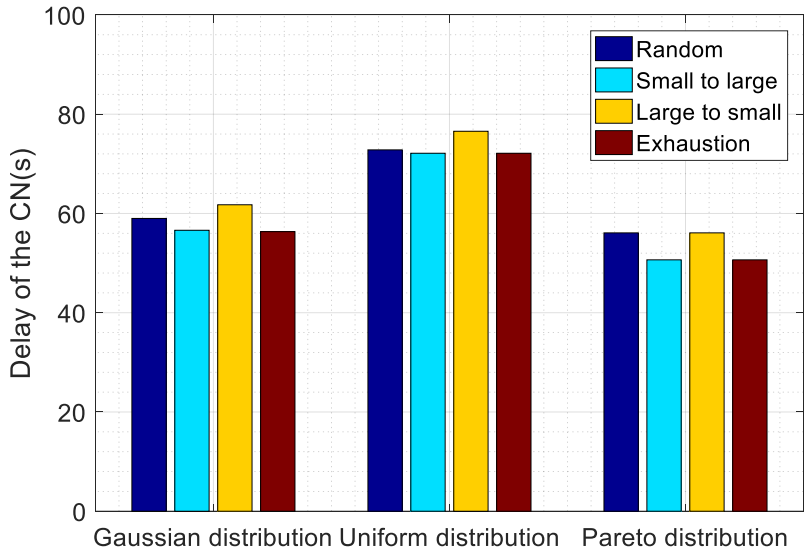

Figure 4. System model. This figure describes the delay of the CN that the computing delay is smaller than communication delay under different task scheduling algorithms. when the node's computing delay is smaller than communication delay, the total delay of the large to small algorithm is close to exhaustion algorithm.

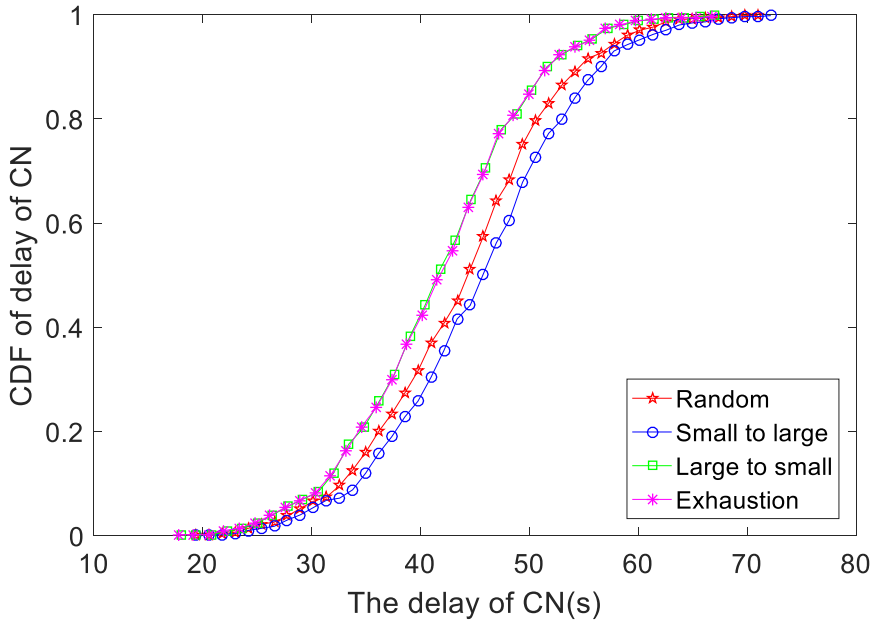

Figure 5. CDF of delay of CN in the scenario communication delay is smaller than computing delay. This figure shows the CDF of the delay of single node under different scheduling algorithms when the size of task follows a gaussian distribution when communication delay is smaller than computing delay. our proposed scheduling algorithm is close to the exhaustive algorithm. This further validate the efficiency of our proposed scheduling algorithm.

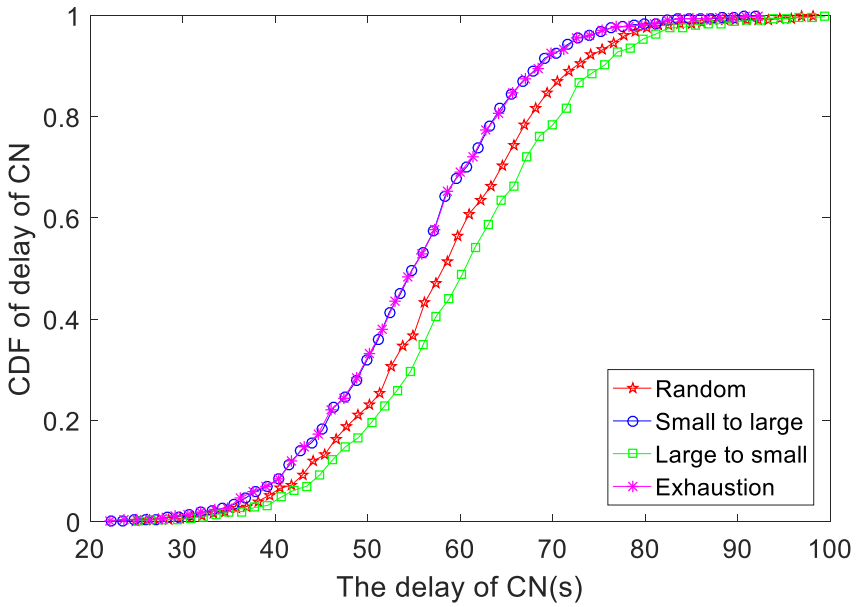

Figure 6. CDF of delay of CN in the scenario computing delay is smaller than communication delay. This figure shows the CDF of the delay of single node under different scheduling algorithms when the size of task follows a gaussian distribution when computing delay is smaller than communication delay. our proposed scheduling algorithm is close to the exhaustive algorithm. This further validate the efficiency of our proposed scheduling algorithm.

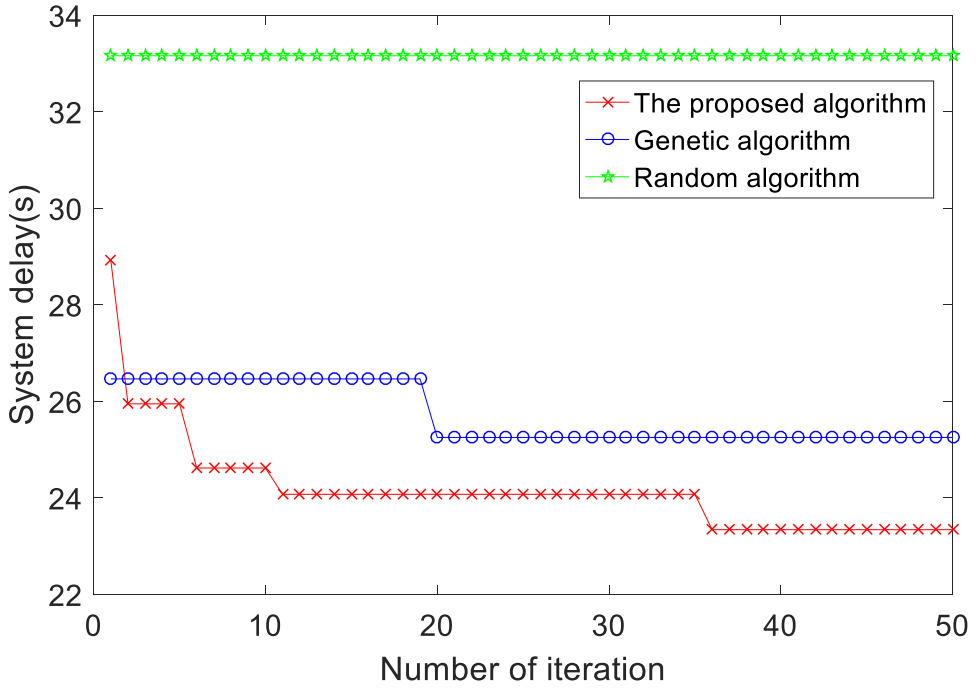

Figure 7. The total system delay under different algorithms. The total system delay is shown in this figure. It can be intuitively seen that the performance of the RA is worst. Compared with the RA, GA can get better performance. But for GA, it is difficult to jump out of the local optimal solution sometimes. But for our proposed algorithm based on the schedule algorithm and firework algorithm, it can get best performance compared with other two algorithm.

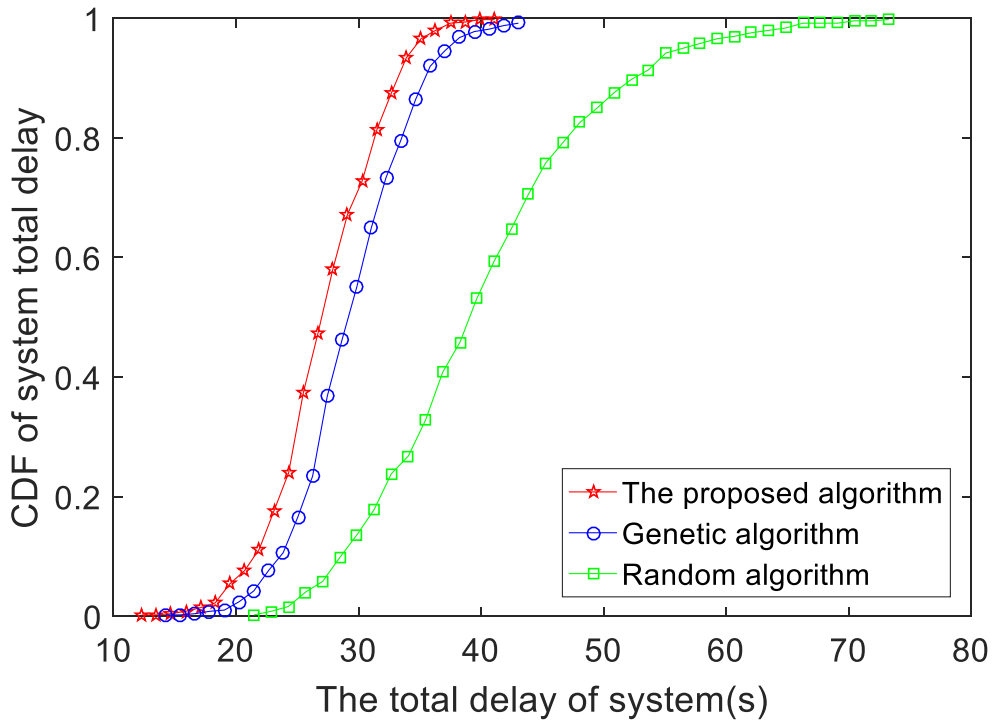

Figure 8. CDF of the total system delay under different algorithms. This figure shows the CDF curves of the total system delay under different algorithms. From this figure, it can be seen that the performance of the RA is

worst due to the randomness of allocation. And the total system delays of these three algorithms are about in range of 11s to 41s. Compared with the RA and GA, the proposed algorithm can reduce the total system delay by 15s and 2s, respectively.

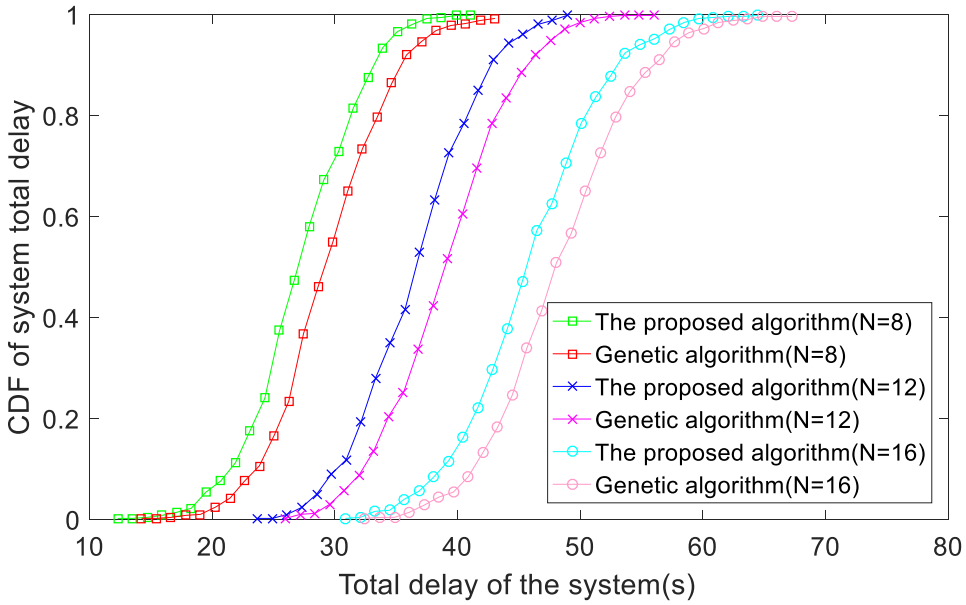

Figure 9. CDF curve of system delay with different number of tasks. This figure shows the relationship between the number of tasks and the total system delay of the GA and our proposed algorithm. It can be observed from the figure that the system delay increases with the increasing of the number of tasks for the two algorithms. Meanwhile, the system delay will increase by about 10s for each additional 4 tasks. Compared with the GA, our proposed offloading algorithm is always better than the GA.

Figures

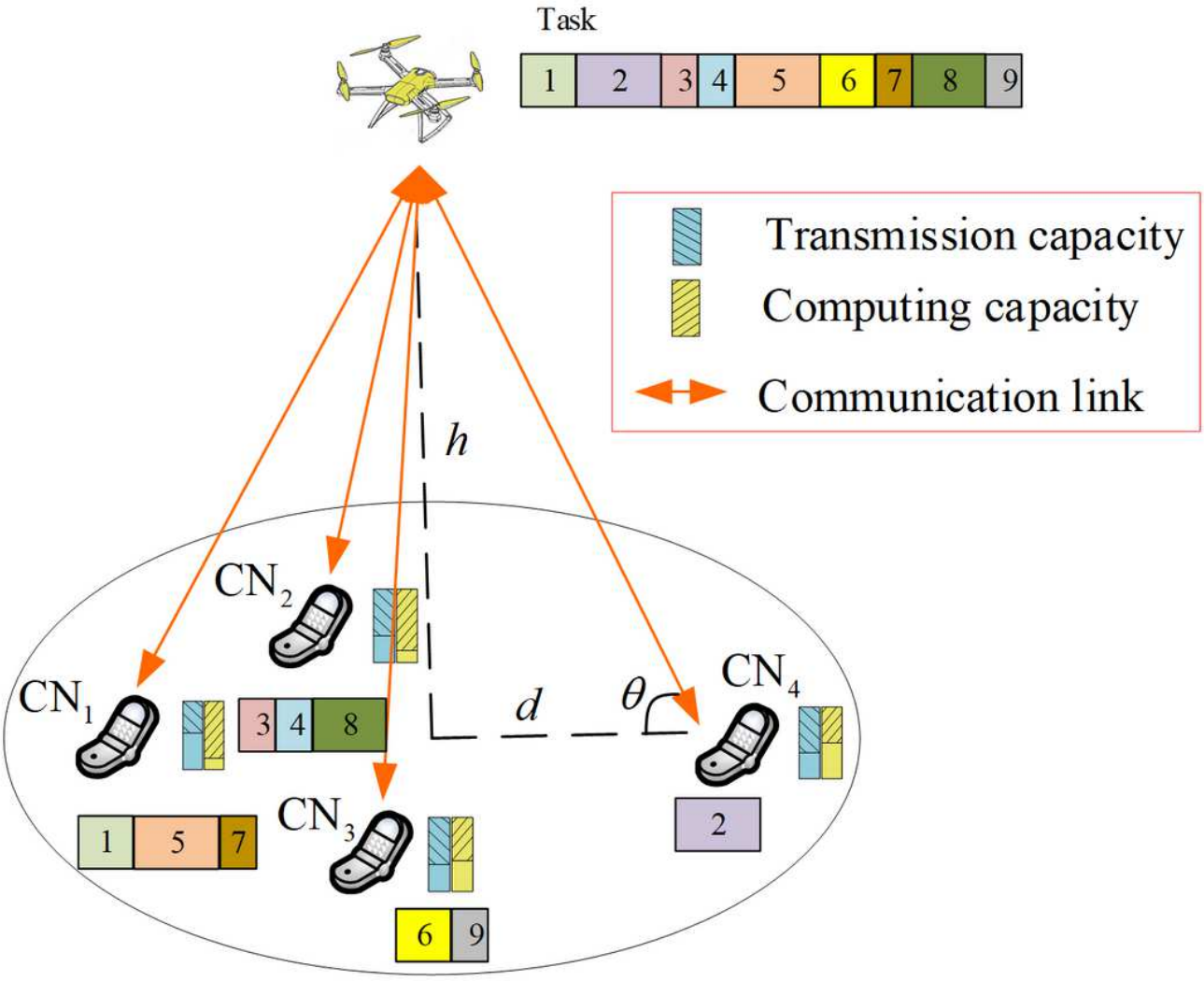

Figure 1

System model

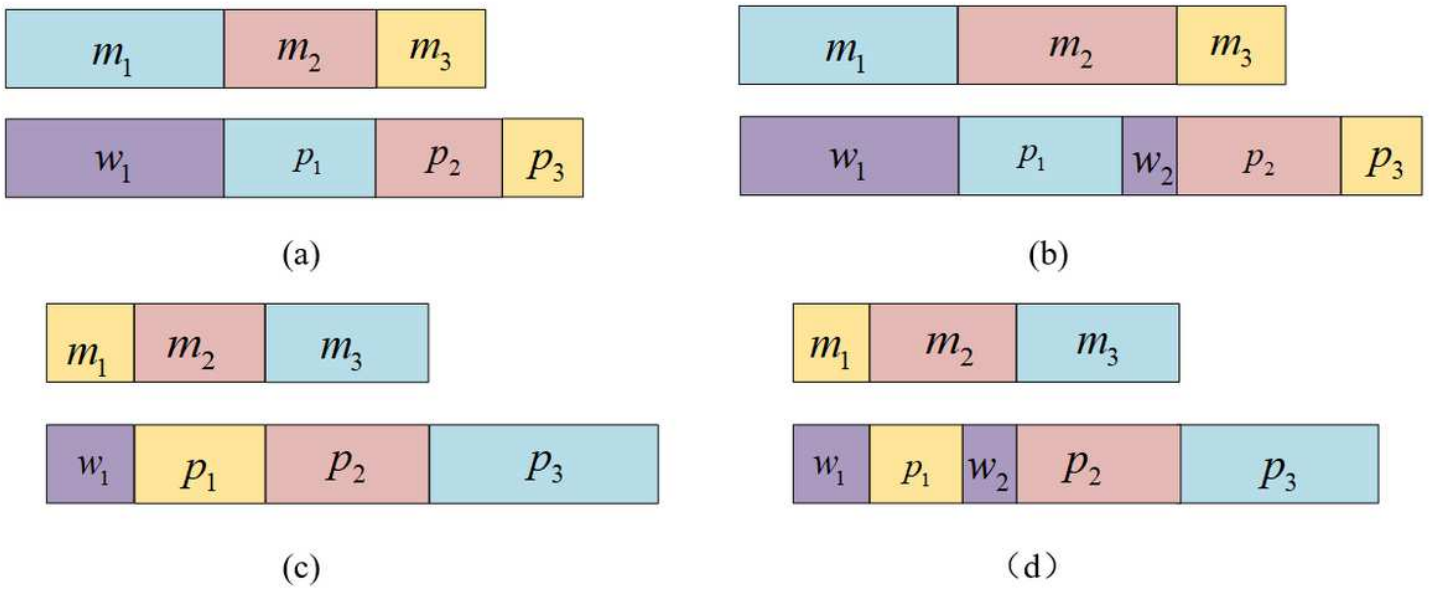

Figure 2

The diagram of delay under different scenarios

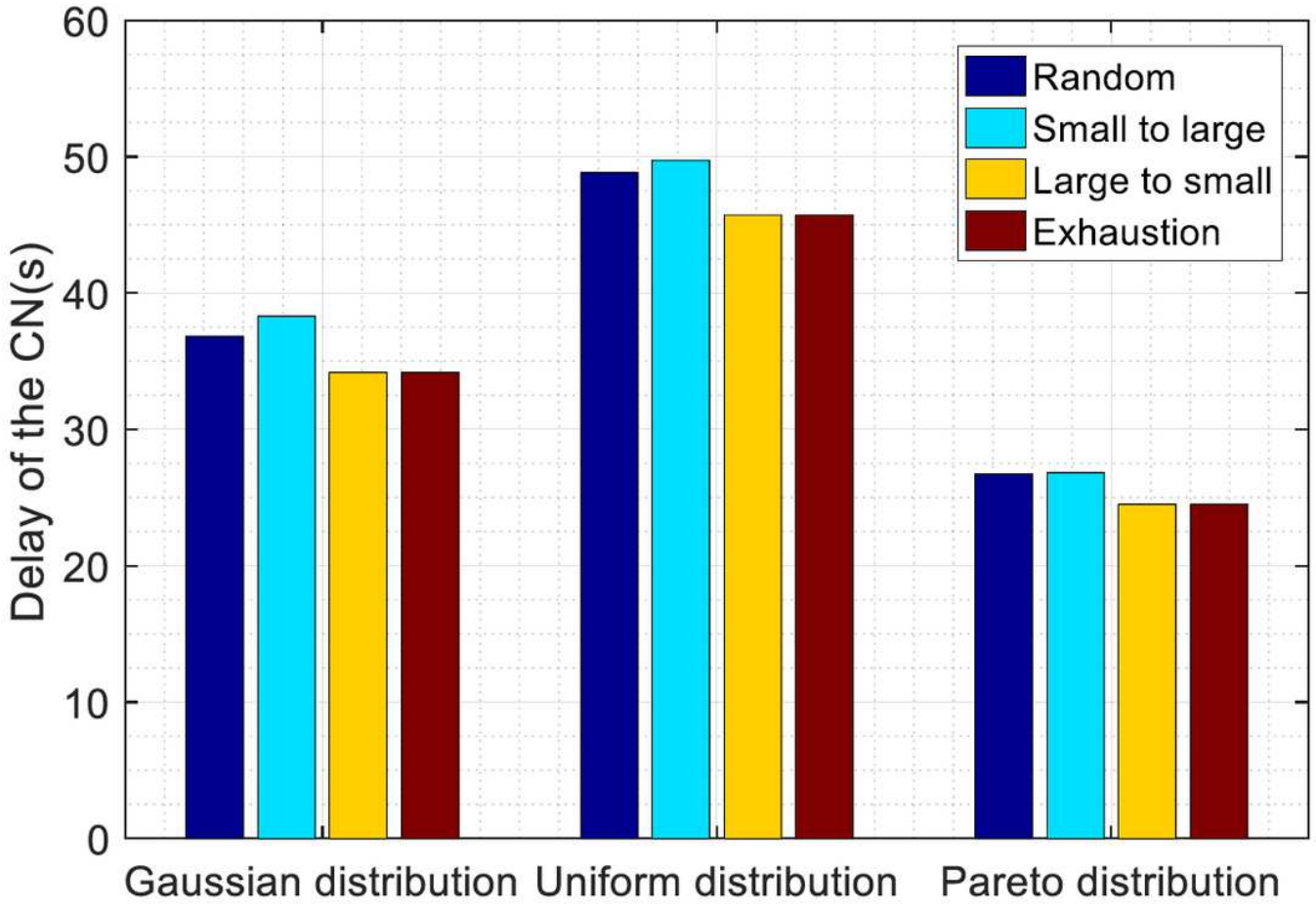

Figure 3

Delay of CN in the scenario communication delay is smaller than computing delay

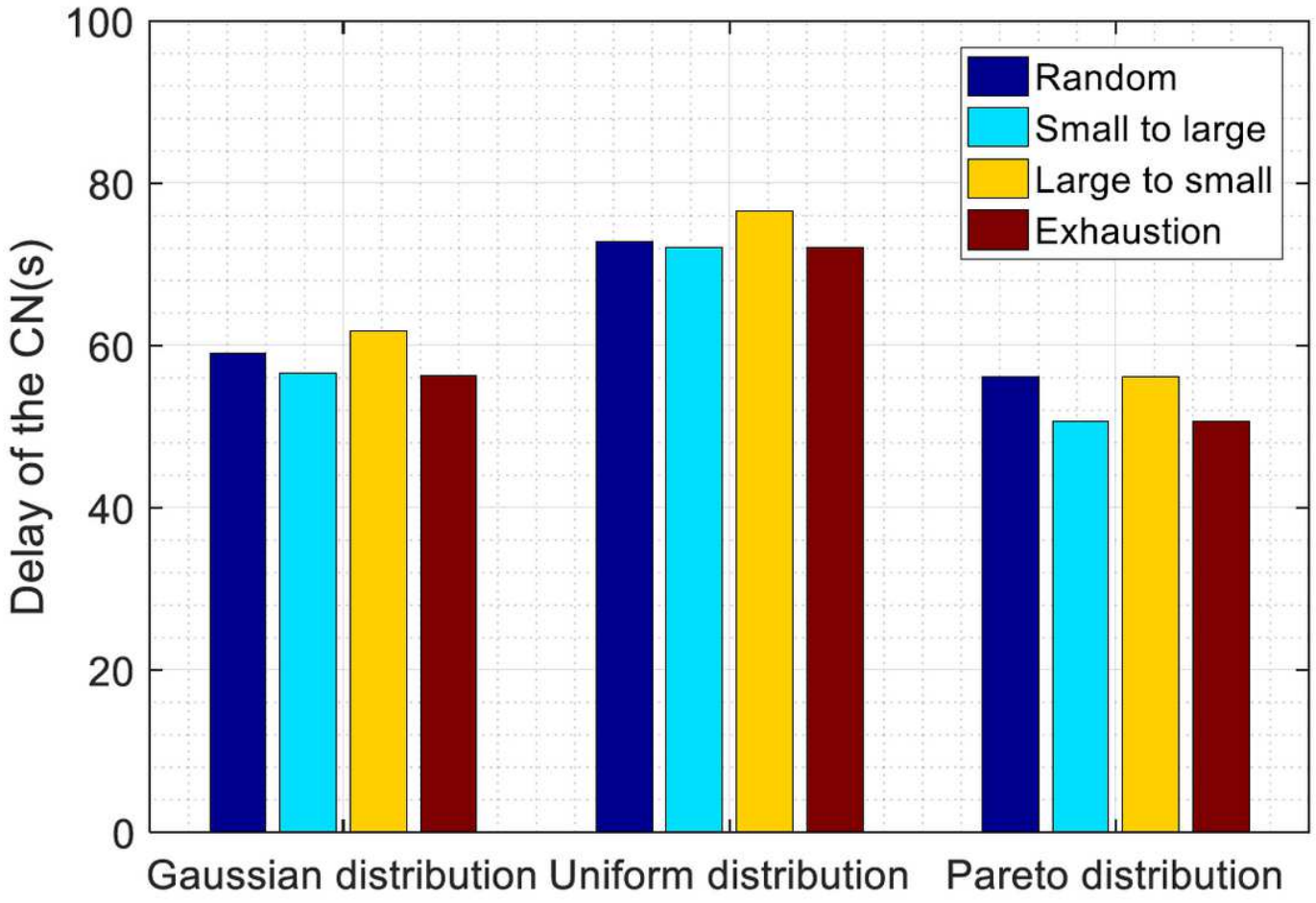

Figure 4

Delay of CN in the scenario computing delay is smaller than communication delay

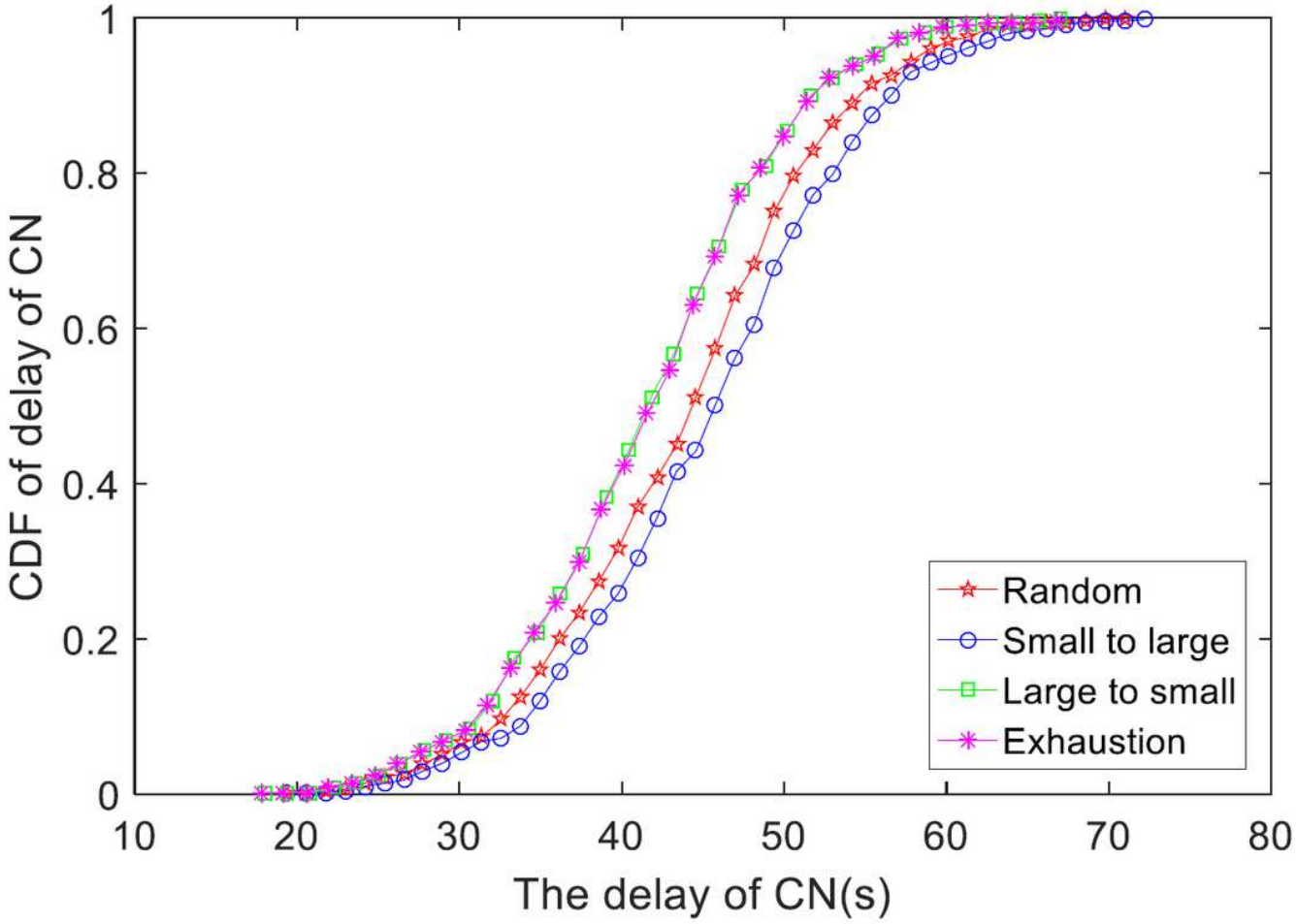

Figure 5

CDF of delay of CN in the scenario communication delay is smaller than computing delay

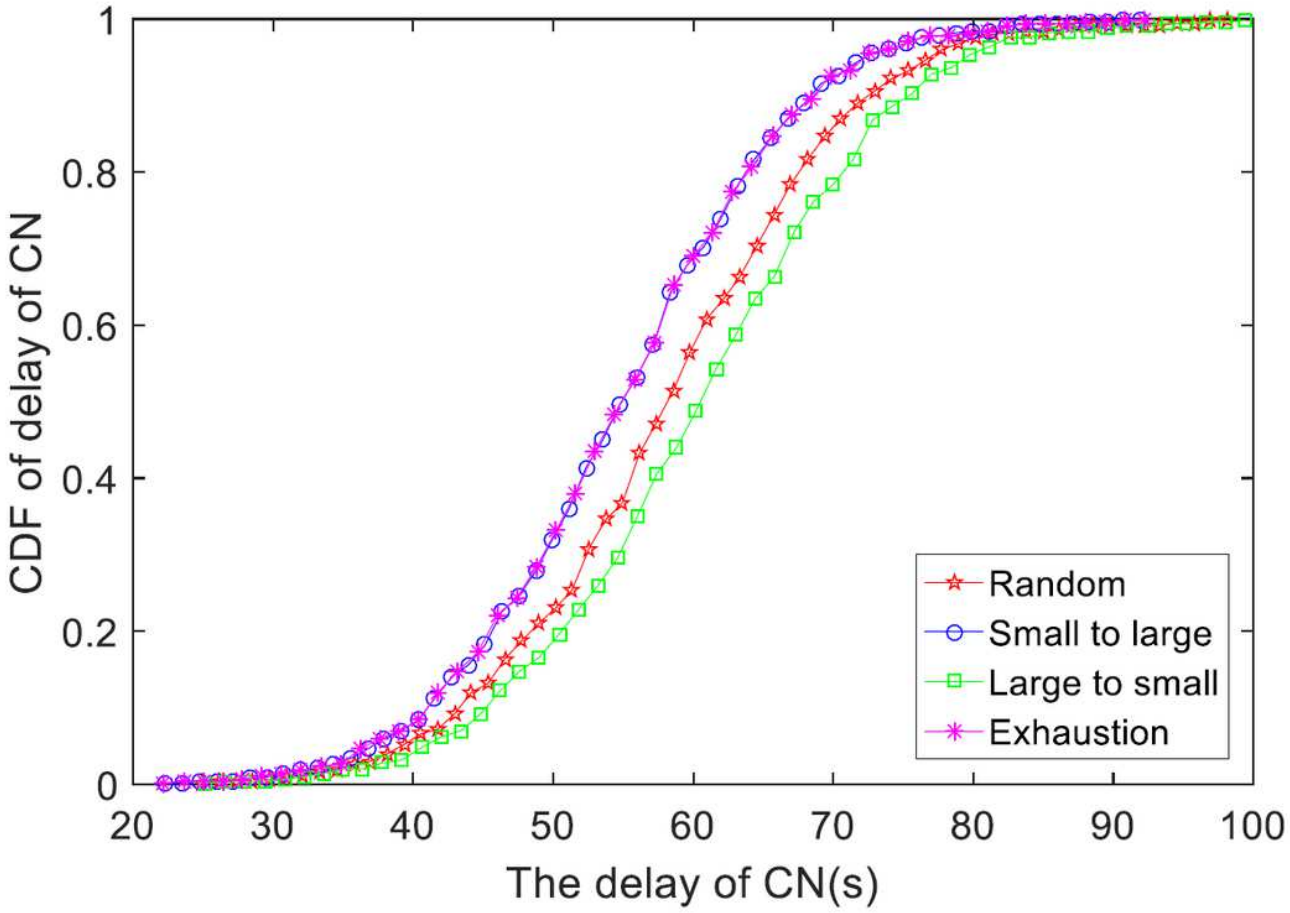

Figure 6

CDF of delay of CN in the scenario computing delay is smaller than communication delay

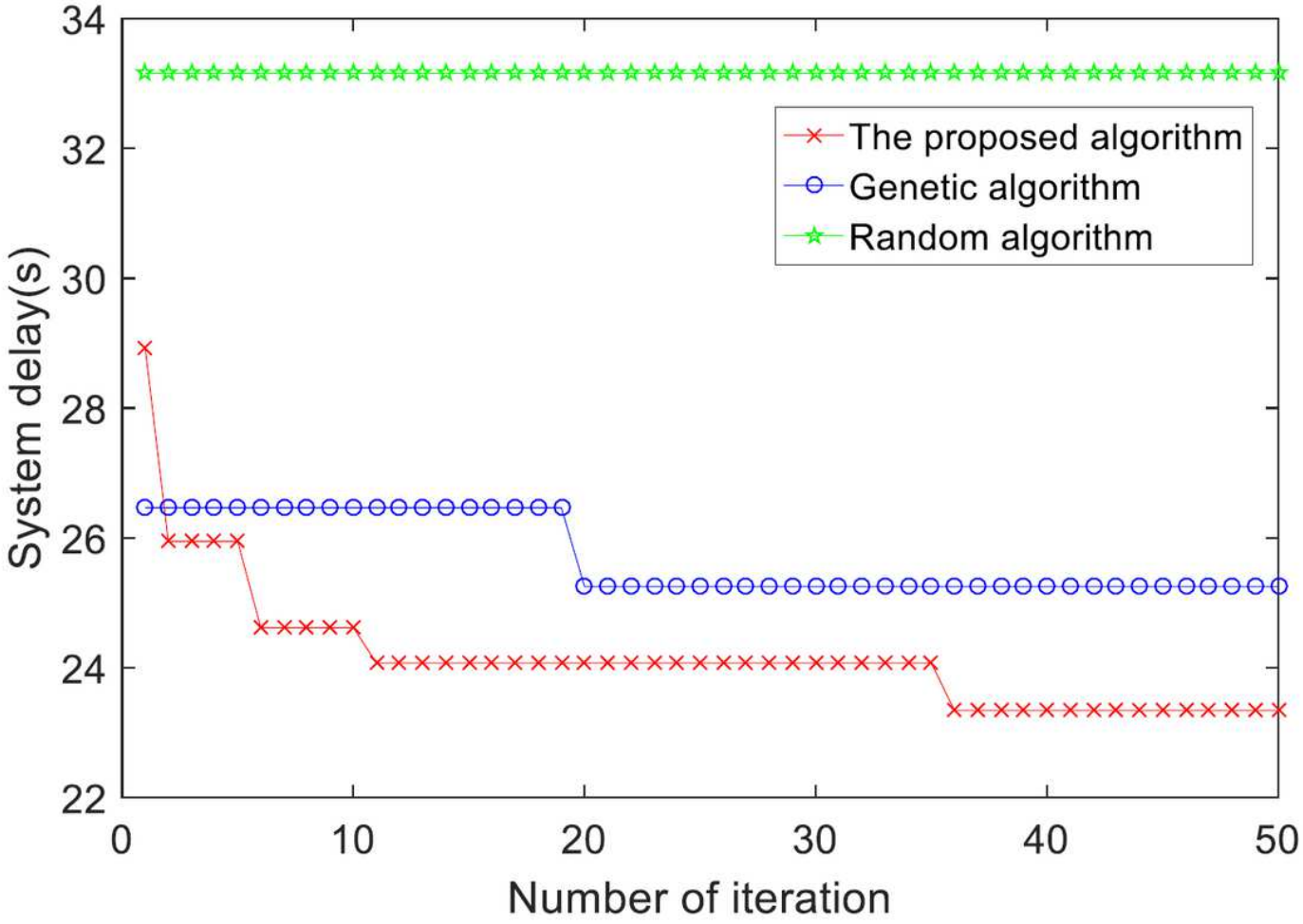

Figure 7

The total system delay under different algorithms

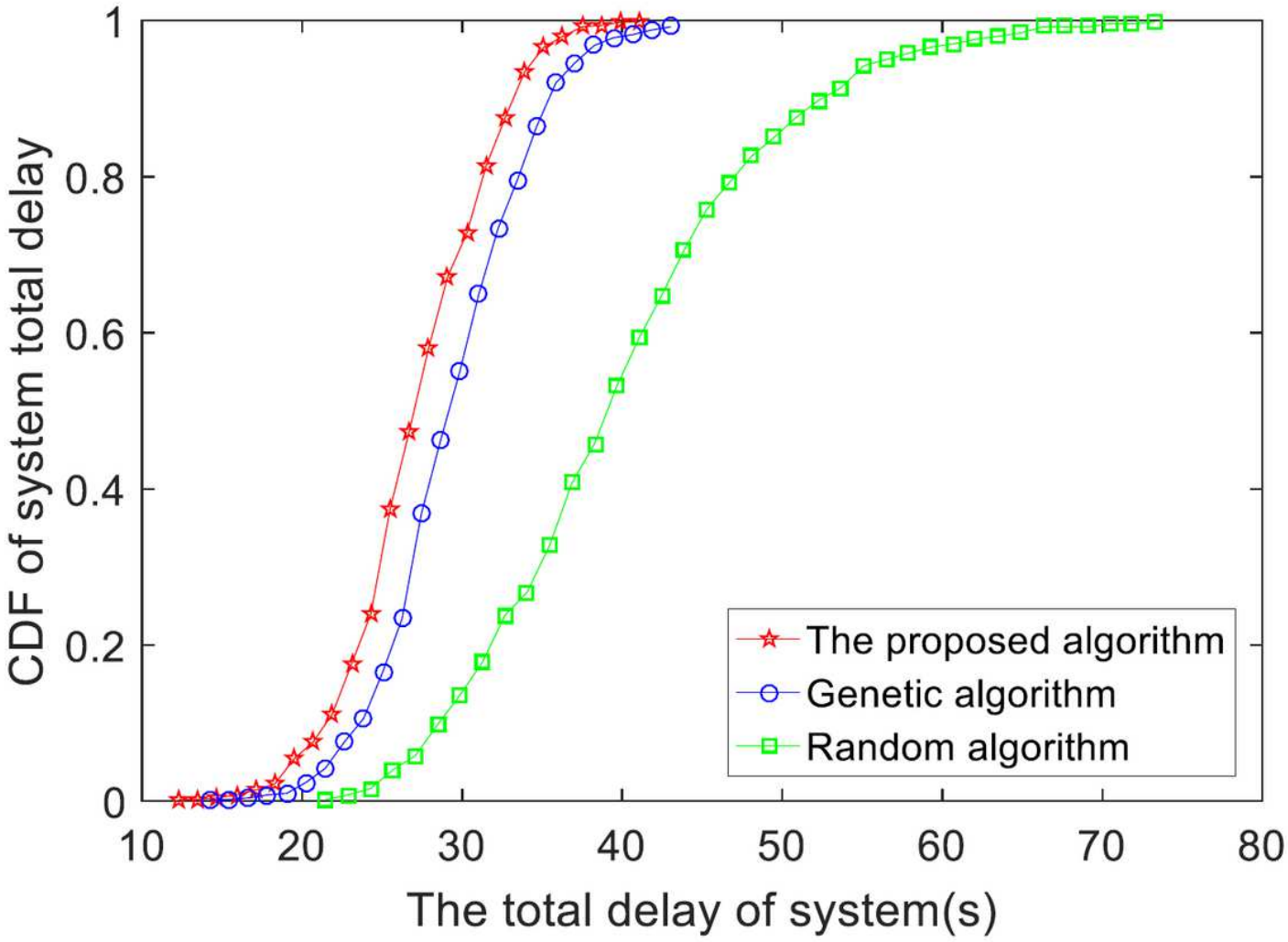

Figure 8

CDF of the total system delay under different algorithms

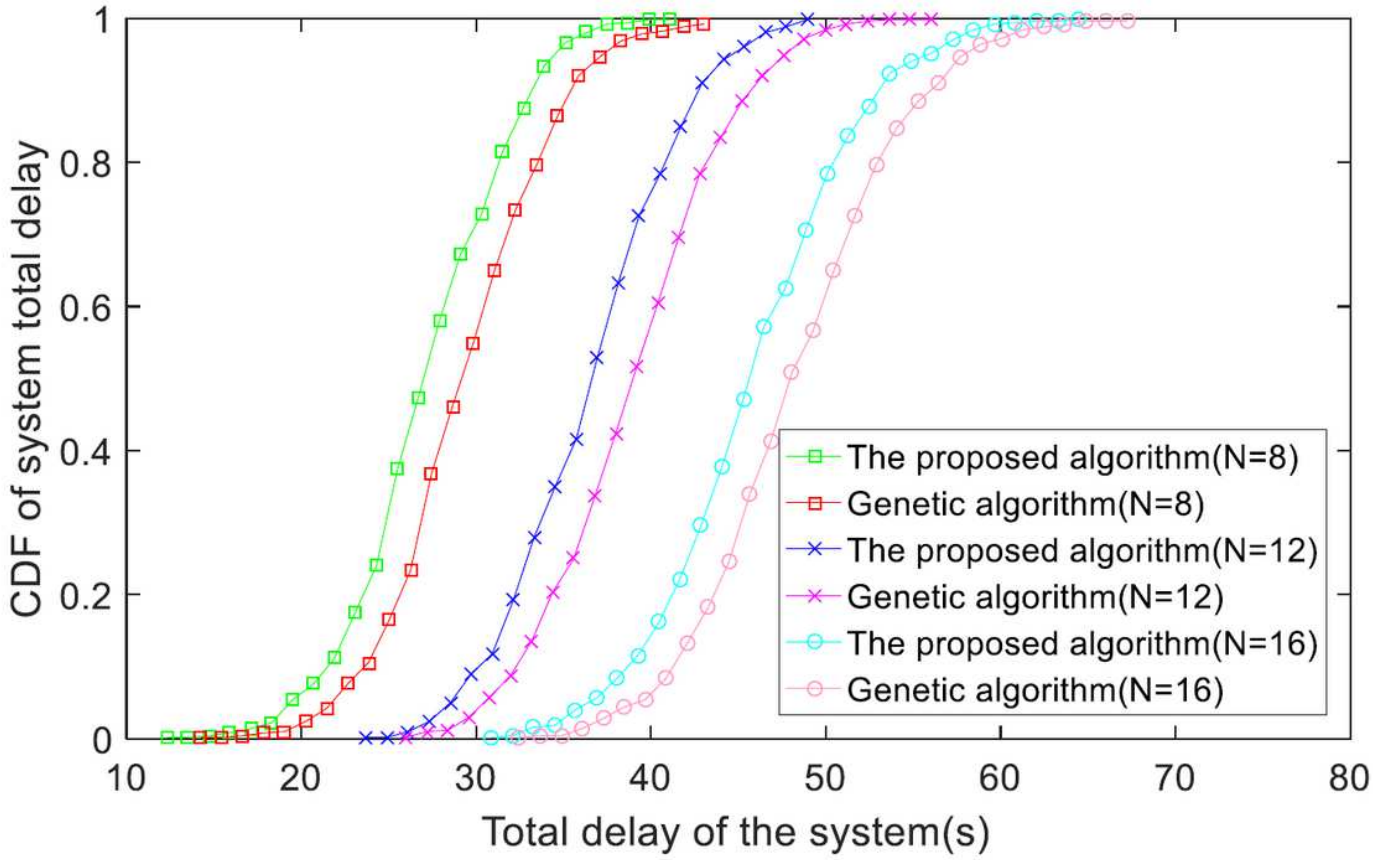

Figure 9

CDF curve of system delay with different number of tasks