Mobile Networks and Applications (2022) 27:236–248
https://doi.org/10.1007/s11036-020-01627-y

Computing Cost Optimization for Multi-BS in MEC by Offloading

Wenzao Li[1,3,5] · Fangxin Wang[2,3] · Yuwen Pan[1] · Lei Zhang[4] · Jiangchuan Liu[3]

Published online: 6 August 2020

Abstract
As today's Internet of Things (IoT) applications are becoming more complicated and intelligent, IoT devices alone can no longer well support the ever-increasing demand for powerful computation and high energy efficiency. Mobile Edge Computing (MEC) and 5G technology emerge as promising solutions, which enable IoT tasks to be offloaded to edge servers for effective processing. Though desirable, there however exists a mismatch between the massive IoT task workloads and limited wireless bandwidth, making it challenging to achieve an optimal offloading strategy at the mobile edge, e.g., the base station (BS) server. In this paper, we aim to migrate the most suitable offloading tasks to fully obtain the benefits of the MEC task offloading. We first formulate the task offloading model as an optimization problem, and theoretically prove the NP-hardness in achieving the optimal solution. Thus, a Genetic algorithm, named M-COGA, is proposed to solve the task offloading selection in both single and multiple BS scenarios. The algorithm focuses on offloading as many tasks as possible with the maximum cost offloading. The proposed cost function takes into account both the computation overhead and energy consumption. Besides, for the multi-BS coverage scenario, we also consider the approach flexibility as well as link load balance. And an enhanced dynamic task offloading scenario is further discussed. We verify the efficiency of our algorithm under the condition of both uniform and non-uniform distribution of covered nodes. Numerical experiments demonstrate that our dynamic allocating scheme can effectively work in MEC offloading. Besides, it largely outperforms the single BS scenarios and reduces the cost of edge devices.

Keywords Mobile edge computing · Task offloading · Genetic algorithm · Computing overhead · Allocating schedule

1 Introduction

1.1 Background and motivation

The mobile Internet and Internet of Things (IoT) applications [1–4] are experiencing an explosive growth in recent years, calling for much higher computation capacity for intelligent processing. Mobile Edge Computing (MEC) emerging as a promising computing allocation scheme [5] enables the interconnected devices to complete many complicated and computation-intensive tasks. Yet, on one hand, these devices mostly suffer from the constrained power supply and limited processing capabilities [6]. On the other hand, an arbitrary strategy to upload local tasks to the cloud will not only increase the burden of cloud but also consume much unnecessary resource. The edge-cloud computing paradigm provides a promising opportunity by offloading the computing tasks of mobile devices to nearby servers (edge clouds) or Base Stations (BSs), which can prominently reduce the computation cost and energy [7]. The limitation of shared wireless bandwidth however restricts the entire offloading tasks, only allowing a portion of tasks to move to the cloud servers [8]. Meanwhile, the limited bandwidth and computation resources also cannot allow all mobile tasks to be offloaded to resourceful BSs. Given that a device is covered by overlapping communication range of multiple BSs, a random task selection strategy will result in inefficient channel utilization and link imbalance.

This is an expanded paper, and the previous version had been published in Qshine 2019.

✉ Fangxin Wang
fangxinw@sfu.ca

Wenzao Li
lwz@cuit.edu.cn

1 College of Communication Engineering, Chengdu University of Information Technology, Chengdu, China

2 Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, Canada

3 School of Computing Science, Simon Fraser University, Burnaby, Canada

4 College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China

5 No.24 Block 1, Xuefu Road, Chengdu, China

Therefore, an efficient strategy is necessary to decide which tasks should be offloaded to proper BSs to achieve higher energy efficiency and more beneficiaries.

1.2 Limitation of prior work

The resource allocation in MEC offloading is one of the key challenges, and there exists many studies for offloading strategy design. In recent works, the formulated problems and optimization objectives of task offloading are different. There are several representative works for further discussion. In the literature of Zhi et al. [9], it focused on the optimization problem of minimizing user task completion time under limited bandwidth resources. And the system model was formulated by Users Devices (UD) and a single BS. Chen at al. [10] formulated several computation decision-making behaviors for devices as a game in MEC. Though they considered the task computation cost between local devices, BS and remote cloud, the selection of BS under the coverage of multiple BSs was ignored. Hao et al. [11] proposed a relatively low-complexity algorithm for energy saving, and the energy efficiency is compared with both local computation and offloading situations. In the paper of Tran et al. [12], it took into account the optimization of energy and time, and it also discussed the resource allocation problem with multi-BSs scenario. But the BS selection strategy and number of uploading tasks are not involved. Other existing researches [13, 14] also proposed a variety of scheduling algorithms that mainly focused on energy efficiency and computation delay. However, these prior works usually ignored the number of migration tasks, which are actually highly dependent on the channel capacity and the task transmission rate. In general, the computation task is offloaded, which indicates that a requirement of an application has been satisfied. To some extent, the higher number of offloading tasks, the more requests will be completed in the network. Hence, the amount of offloading tasks is related to the user's experience. At the same time, most of them also ignored considering the multi-BS selection scenario, which can render imbalanced task loads for different BSs. In particular, we are also interested in the problem of Multi-BS offloading scenarios and BS load balance in the case of unevenly distributed terminals. In order to further explain the differences in current solutions, we compare state-of-the-art solutions as presented in Table 1.

Therefore, the multi-BS selection methods with limited channel capacity are valuable to discuss in MEC.

1.3 Contributions and organization

There are several key challenges in our task offloading selection scenario. First, mobile devices are usually densely distributed and covered by more than one BS. An improper BS selecting scheme will result in imbalanced workload and poor offloading performance for the whole system. Besides, given the massive number of terminal devices in practical applications, a model with exceptional scalability and versatility is required. Moreover, a practical task offloading system needs to consider multiple optimization objectives such as efficiency and energy. It is getting even more complicated to simultaneously integrate multiple objectives together to achieve a terrific offloading selection strategy.

In this paper, we formulate the task offloading model as an optimization problem, and theoretically prove the NP-hardness to achieve the optimal solution. Besides, a BS selection scheme is taken into account to expand the data capability in the overlapping coverage scenarios. A Multi-BS based Genetic Algorithm named M-COGA is proposed to solve the task offloading selection in both single and multiple BS scenarios. The algorithm focuses on offloading as many tasks as possible with the maximum cost. And this cost includes energy consumption and computation time for local computing tasks. The number of offloading tasks is jointly considered in our design purpose. Furthermore, the proposed M-COGA algorithm is optimized in mutation operation according to the modeling characteristics of task offloading. From the simulation results, the optimized M-COGA algorithm has fewer iterations. Besides, for the multi-BS coverage scenario, we also consider the approach flexibility as well as link load balance. And an enhanced dynamic tasks offloading algorithm is further proposed. We verify the efficiency of our algorithm under the condition of both uniform and non-uniform distribution of covered nodes. In general, our algorithm achieves superior performance in task selection, computation cost offloading, BS load balancing, and energy efficiency. Besides, the algorithm in the dynamic task offloading model also greatly improves the task migration efficiency in the ultra density networks. Numerical evaluations demonstrate the superiority of our solution.

The rest of this paper is organized as follows. The computation offloading system model is presented in Section 2. Section 3 introduces the multiple BS system model. Then, we propose a genetic algorithm and the application under dynamic task offloading scenario in Section 4. In Section 5, we present the numerical experiments and analyze the results to evaluate the performance. We finally discuss the related work in Section 6, and conclude paper in Section 7.

2 Computation offloading system model

Computation offloading is usually discussed in different scenarios, and the considered factors are inconsistent in different system models [5, 10]. Given that those devices

Table 1 The comparison of several task offloading algorithms

Literature	Two layer topology	Three layer topology	Time/Energy resource	Single BS scenario	Multi-BSs scenario	Number of uploading tasks
Zhi et al. [9]	Y	-	Y	Y	-	-
Chen et al. [10]	-	Y	Y	Y	-	-
Tran et al. [12]	Y	-	Y	-	Y	-
M-COGA	Y	-	Y	Y	Y	Y

are all communicable to one or more BSs, carrying out the offloading strategy at BSs is a simple way to reduce the terminal cost. In most cases, the terminal is covered by multiple BS. We also consider a terminal set $\mathcal{U} = \{u_1^{(\mathcal{B})}, u_2^{(\mathcal{B})}, ..., u_i^{(\mathcal{B})}, ..., u_k^{(\mathcal{B})}\}$ as the covered terminals by BS set $\mathcal{B}$, which $\mathcal{B} = \{\mathcal{B}_1, \mathcal{B}_2, ..., \mathcal{B}_j\}$. Generally, the j usually is a smaller number, it means that these devices are covered by one or small number of BSs, which can provide the wireless channel to the set $\mathcal{U}$. We assume that the set $\mathcal{U}$ will not be changed during the offloading period $\mathcal{T}$, and each device has one task offloading requirement. Generally, the number of devices and channel gain may be changed due to the mobility of users covered by the base station. Each device may be owns more than one task, which needs offloading to the remote side. This mode of task generating task is not discussed in this paper, and it will be further discussed in future works. Therefore, the task set $\mathcal{T}_{task} = \{L_1, L_2, ..., L_i, ..., L_k\}$, where $L_i^{(\mathcal{B}_j)} \in u_i, \mathcal{B}$. Each covered device only has one task in the model and the task of the device is required to offload to a near BS by centralized control of itself. In some scenarios, we need to sort these multiple tasks due to bandwidth constraints or task priority. And the simplest strategy is only uploading one task to the MEC server at the moment. As shown in Fig. 1, the task offloading scenario has four important parts, including the offloading tasks, the wireless channel, the local users, and the remote edge cloud. In this proposed system model, we focus on the computation task between terminal and BSs.

The task offloading schedule can reduce the computing overhead and energy consumption of terminals. Therefore, the communication mode, mobile device and cloud play as pivotal roles in the MEC. These models are introduced in detail as follows.

2.1 Communication model

In the OFDMA communication system, the total bandwidth is partitioned into several sub-channels. And the OFDMA-based cellular network usually adopts the full-duplex (FD) ratio technology and FD based BS supports multiple half-duplex (HD) users [15]. The numerous sub-carriers of each sub-channel can be assigned to a user in a centralized model. Assume that U means the uploaded bandwidth, and the $g_{(U)}$ represents the channel gain for a mobile, the channel bandwidth is B, and the transmission power is S_U. The N_0 presents the average noise power, it is usually considered as Gaussian channel noise. Then maximum transmission bit rate C_U can be calculated by equation (1)

$$C_U = B * log_2(1 + \frac{g_U S_U}{N_0}), \tag{1}$$

where C_U represents the maximum data rate. It means the maximum data transfer rate can be provided to BS covered users. But it is restrained by some specific parameters of BS. In actual situations, the data transfer rate $C_S(U)$ is largely below the theoretical value of C_U. Although there is data conflicting in the same spectrum, it can be improved by the physic layer channel access schedule such as CS-MUD and SCMA [16, 17]. From the equation (1), the limited transfer data rate may not satisfy the unlimited number of mobile devices to offload to the BS. For the reason of task processing efficiency, the more tasks benefited from the remote cloud, the better in MEC scenarios.

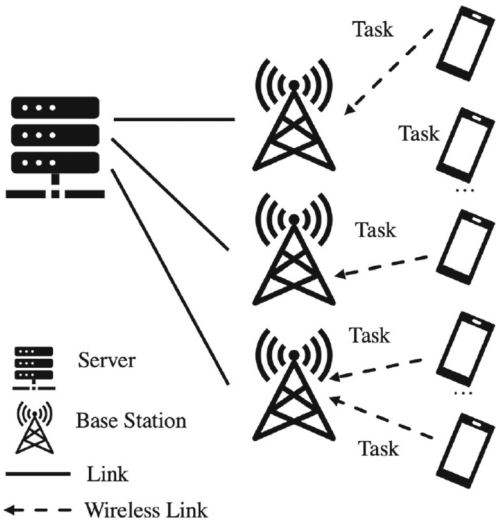


Fig. 1 The task offloading in multi-user MEC system

2.2 Mobile device computation model in task computation offloading

The energy consumption and computing time in local device are frequently discussed [10, 18], we assume that device i has only one task $L_i \triangleq (d_i, f_i^l)$, the f_i^l denotes the CPU cycles of mobile devices per second. Then the local computing time can be described as (2)

$$t_i^l = \frac{d_i}{f_i^l}, \tag{2}$$

where t_i^l can also be understood as execution time or the execution cost of a terminal user. Then, the energy consumption can be described as (3)

$$e_i^l = \mathcal{J}_i d_i, \tag{3}$$

where $\mathcal{J}_i$ denotes the coefficient consumed energy per CPU cycle. If we transform this problem as the energy consumed per bit of CPU processing, we can describe the energy consumption as equation (4)

$$\eta_i^l = \frac{\mathcal{P}_i t_i^l}{\mathcal{D}_i}, \tag{4}$$

where $\mathcal{P}_i$ represents the CPU power consumption. Let t_i denotes the processing time of task L_i. From equation (3) and (4), the processed amount of data or the number of CPU cycles for the task are proportional to the energy consumption. Note that the CPU keeps the computing frequency in the processing period and it is difficult to accurately calculate the energy consumption per cycle due to the complicated work model of a CPU. Hence, $\mathcal{J}_i$ can be understood as a coefficient of energy computing. As mentioned above, the task is offloaded to the nearby BS to obtain relatively abundant computing resources. The offloading purpose is meeting a requirement of an application, and the partial offloading of a task can not return the computation results of the users. And we consider the data that a task needs to upload as a whole. After we establish the computing time and energy consumption model with (2) and (3), then task cost of the local user u_i can be defined as (5)

$$\mathcal{C}_i^l = \lambda_t t_i^l + \lambda_e (e_i^l + P_r), \tag{5}$$

where λ_t and λ_e denote the weight parameters that influence the optimization target for the concerned network indicators. P_r denotes the task priority value. Because the value of e_i^l is an assumed value in the validation protocol, and P_r is positively correlated to the task cost of local user. Then, we will not set the value in the next simulations. From equation (5), if the system cares about the energy consumption, then it can set $\lambda_t < \lambda_e$ and $\lambda_t, \lambda_e \in [0, 1]$. It is the common processing way in the weight method.

2.3 Cloud computing model

The remote cloud is considered to have sufficient computing ability, and the computing energy has no constraint due to the power supply. Some studies focus on the overall operation to reach a reasonable balanced state. For instance, the task offloading needs to satisfy $\mathcal{C}_i^l < \mathcal{C}_i^c$, where the $\mathcal{C}_i^c$ denotes the task computing cost in the cloud [10]. Some other studies do not only consider the computing capability of the remote cloud, but they also consider the capacity of the wireless channel. As shown in subsection 2.1, the channel capacity is limited by C_U and there are $\mathcal{M}$ sub-channels. Their respective bandwidths make up the available bandwidth B. To get the benefits of the cloud servers through the limited bandwidth is a challenging question. However, we focus on the transmission bandwidth between terminals and BSs, and the bandwidth between the BS and the cloud is sufficient. Therefore, the paper does not consider the computing power and cost of the cloud.

3 Distributed computation offloading in multiple BSs

The proposed computation offloading has been discussed in this section. As represented in the prior section, a sub-channel is used by user i. Therefore, the channel capacity of the optional $\mathcal{B}$ can reach the maximum data rate (6):

$$C_{\bar{s}}^{(\mathcal{B})}(i) = \lambda_\gamma \bar{B} * log_2(1 + \frac{g_i S_i}{N_0}). \tag{6}$$

The $\bar{B}$ represents the bandwidth of a sub-channel and λ_γ can be understood as channel utilization ratio. Then we can formulated the channel data rate C_U as $C_U \geq \sum_{m=1}^{n} L_m C_{\bar{s}}(m)$. The $L_m \in \{0, 1\}$ denotes whether it is the determined offloading tasks, and $C_{\bar{s}}(m)$ represents the channel ratio of task m. In the MEC scenarios, the bandwidth allocated to offloading tasks is limited. Thus, it determines that the uploaded data rate cannot be greater than the data rate under the total allocated bandwidth in a time period. Therefore, under the condition of C_U, we want to get the number of uploaded tasks as much as possible. Besides, we want to reduce the most energy cost and computation cost of tasks. So the resource allocation problem under the

constraints on the channel data rate can be formulated as follows:

$$\begin{aligned}\mathcal{Z}_{\mathcal{N}_m,\mathcal{V}_c} &= \begin{pmatrix}\max\limits_{\mathcal{N}_m} f(\mathcal{N}_m)\\ \max\limits_{\mathcal{V}_c} f(\mathcal{V}_c)\end{pmatrix}\\ &= \begin{pmatrix}\max\limits_{\mathcal{N}_m} \sum_{m=1}^{n} L_m * C_{\bar{s}}(m)\\ \max\limits_{\mathcal{V}_c} \sum_{m=1}^{n} L_m * \mathcal{C}_i^l(m)\end{pmatrix}\\ s.t.\quad & C_U \geq \sum_{m=1}^{n} L_m C_{\bar{s}}(m)\\ & L_i \leq N_t, i \in N_t, C_i \leq C_{\bar{s}}(i)\end{aligned} \tag{7}$$

Equation (7) describes the demand of task offloading. The purpose of the scenario requires that maximum tasks $\mathcal{N}_m$ which need to offload to the cloud. Unfortunately, it should face the limited bandwidth. And the compute offloading still needs to consider the energy efficiency $\sum_{i=1}^{m} e_i^l$ and the compute time $\sum_{i=1}^{m} t_i^l$. In the model, cloud computing capabilities are considered to be sufficient. Both the energy cost and computing time require maximum value $\mathcal{V}_c$ to reduce the burden. Obviously, this is a multi-objective optimization problem. And unfortunately, it is extremely challenging to obtain an optimal solution.

Theorem 1: The problem of finding the maximum number of offloading tasks as presented in (7) is NP-hard.

Proof: It is well known that the classical knapsack problem is defined as follows. There is a set of n items, where each item i has a value p_i and a weight w_i. The capacity of the bag is represented as c. The problem can be formulated as finding the maximum profit with a set of items. This model can be described as

$$\begin{aligned}&\max \sum_{i=1}^{n} p_i x_i\\ &s.t.\quad \sum_{i=1}^{n} w_i x_i \leq c\\ &x_i \in 0, 1,\ j = 0, 1, ..., n.\end{aligned} \tag{8}$$

where x_i means whether the item is in the knapsack or not. And it is well known as an NP-hard problem [19]. Turning back to our task offloading problem, the $C_U \geq \sum_{m=1}^{n} L_m(m) C_{\bar{s}}(m)$. If we consider the cost $\mathcal{C}_i^l$ as the value of item and the $C_{\bar{s}}(m)$ is thought as the weight, the capacity of bags can be considered as the limited channel bandwidth. The formulated resource allocation problem also takes into account the maximization of the number of offloading tasks. Thus, if the number of uploading tasks is ignored, this problem can be reduced to a case of the knapsack problem. We therefore prove that our problem is NP-hard.

Then, we need to make the decision of computation offloading to decide which tasks should be offloaded. In a similar way, the selected tasks should have the maximum cost. In this way, this strategy will minimize the overall burden of the terminals.

4 The genetic algorithm for tasks offloading

In this section, the proposed optimization problem is formulated as the equation (7), and the offloading schedule is designed with a heuristic search method. To solve this NP-hard problem, this proposed genetic algorithm focuses on closing the optimal performance.

4.1 Initialization model

First, a matrix In is given to express the decision of computation offloading. The number of rows in the matrix represents the number of offloading decision combinations.

$$In = \begin{bmatrix}\delta_{1,1} & \delta_{1,2} & \ldots & \delta_{1,k}\\ \delta_{2,1} & \delta_{2,2} & \ldots & \delta_{2,k}\\ \ldots & \ldots & \ldots & \ldots\\ \delta_{m,1} & \delta_{m,2} & \ldots & \delta_{m,k}\end{bmatrix}, \tag{9}$$

where the δ_{ij} means the task number j, and each matrix row represents the offloading task order form set $\mathcal{T}_{task}$. The matrix is established by a random way at the beginning, each task index is unique and $\delta_{ij} \in [1, k]$. Besides, the initial matrix is executed in each period t_p. The size of the row is $n = k$, which denotes the number of pending offloading tasks in the BS. And the fitness function $\psi_f(\bar{N}_t, \mathcal{C}_i^l)$ needs to be given according to the optimized object

$$\psi_f(\bar{N}_t, \mathcal{C}_i^l) = \lambda_{\bar{N}_t}^f M(\bar{N}_t) + \lambda_{\mathcal{C}}^f M(\mathcal{C}_i^l), \tag{10}$$

where the $\lambda_{\bar{N}_t}^f$ and $\lambda_{\mathcal{C}}^f$ denotes the coefficient separately. The M function is a mapping function, which can solve the problem of adding non-similar physical dimensions. They are mapped in $[0, \delta]$, then it can be compared within a quantified range. And $\bar{N}_t$ represents the determined offloading tasks, and it takes values from matrix row $\delta_{i,...}$. Besides, $\mathcal{C}_i^l$ means the cost value of determined offloading tasks, which has been described in the equation (5).

4.2 The tasks selection processing by M-COGA

First of all, a terminal needs to determine the set $\mathcal{B}$ which the terminal can communicate with. Then, this terminal will request the remaining bandwidth $\widetilde{C}_{\bar{s}}^{(\mathcal{B})}$ with the BS set $\mathcal{B}$. And the BS with the maximum value of channel capacity $\widetilde{C}_{\bar{s}}^{(i)}$ will be selected. Then, M-COGA establishes the matrix with random values. Then, the M-COGA conducts the

following four phases of operation: roulette algorithm, elite retention strategy, cross operation, and mutation operation.

Roulette Algorithm (RA): RA can be described as roulette wheel selection strategy in GA. In RA, there are three steps for matrix reorganization. First, the fitness value of matrix In should be calculated as the set $f^v(i = 1, 2, 3..., m)$. Second, the survival probability $p_r(r_i)$ is calculated by equation (11)

$$p_r(r_i) = \frac{f^v(r_i)}{\sum_{j=1}^{m} f^v(r_j)}, \tag{11}$$

After we calculated the $p_r(r_i)$, we can get a vector of one column m rows. It represents the probability that each row of tasks can be selected for the next step. Third, $p_r(r_i)$ is used to select rows based on a virtual roulette. The row with a relatively high survival probability will be selected multiple times, and the selection processing ends after the vector $p_r(r_i)$ traversal is complete. Then the m times selecting operations will establish a new matrix In_r.

Elite Retention Strategy (ERS): The ERS just selects a maximum fitness value of a row in In_r, the *best* row is kept at the row $m + 1$. Thus, a new matrix In_e is established.

Cross Operation (CO): M-COGA randomly selects two rows of In_e (except the $m + 1$ row) for *crossing operation*. And it generates an index p, and $p \in \{p \in N | 1 \leq p \leq N_t\}$, then the two rows will change the values before $\delta_{p,...}$ between the two rows with distance one or two position. Besides, a crossing probability P_c, which is introduced in [20], can be described as (12)

$$P_c = \begin{cases} \max(P_c) - \frac{\max(P_c) - \min(P_c) * (\alpha - \beta)}{\delta' - \beta}, \alpha > \beta, \\ \max(P_c) \qquad\qquad\qquad\qquad , \alpha \leq \beta \end{cases} \tag{12}$$

where $max(P_c)$ denotes the the maximum cross probability, and the $min(P_c)$ means the minimum cross probability. α represents the maximum fitness value of the two selected rows (the two orders of offloading decisions). The β denotes the average of calculated fitness value for the whole matrix, and the δ' denotes the maximum calculated fitness value for the matrix In_e. If a randomly variable seed $S_d < P_c$, after CO, the new matrix In_c is established. It should be noted here that the task index is unique in each row after CO. It can be achieved by traversing the task index.

Mutation Operation (MO): After the CO, we can get the new matrix In_c. The M-COGA operates the MO for each row of matrix In_c. First of all, the mutation probability [20] P_m can be described as (13)

$$P_m = \begin{cases} \max(P_m) - \frac{\max(P_m) - \min(P_m) * (\alpha' - \beta)}{\delta' - \beta}, \alpha' > \beta, \\ \max(P_m) \qquad\qquad\qquad\qquad , \alpha' \leq \beta \end{cases} \tag{13}$$

The variables are similar to the equation (12). The $max(P_m)$ represents the maximum mutation probability and the $min(P_m)$ represents the minimum mutation probability. The α' represents the fitness value of current MO row. The $max(P_c)$, $min(P_c)$ are set as 0.9 percent and 0.6 percent, respectively. Besides, $max(P_m)$, $min(P_m)$ are set as 0.9 percent and 0.6 percent, respectively. Each row of In_c will process the MO by the probability P_m. If a row is determined to be mutated, randomly swap the positions of the two index in the row. Such a MO iterates through the first m rows of the entire matrix In_c.

After the MO, the row with the lowest fitness value will be deleted. Then, the new matrix In_m becomes $m * k$. The COGA will iterate until the fitness value is stable.

In the CO stage, for the dramatic changes of each row, the cross operation does not just run once and it is related to the number of initial tasks. Hence, the CO times $\phi(m)$ is designed as (14)

$$\phi(m) = \frac{m}{\lambda_b + \lambda_g \delta}, \tag{14}$$

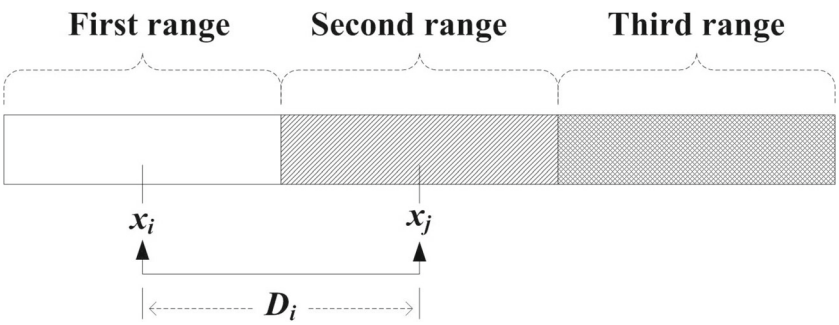


Fig. 2 The mutation distance and the range division for a task row

where λ_b and λ_g are constant coefficients, and the value of $\phi(m)$ is positively related to m. We present δ as a random seed. Then, the more dramatic CO operation will lead to less convergence consumption times.

Algorithm 1 Multiple Base stations based Computing Offloading with Genetic Algorithm.

```
0 Base station selection: Maxmum C̄_a^(B)
1 Initialization: establish the matrix In=N_t × m
2 RA and ERS:
3 Find the biggest fitness value of In and add it to
the rear of matrix In as In_r, In_r=N_t × (m + 1)
4 Repeat m times
5   generate random flags and pro=0
    for repeat m times
6     pro += pro for accumulated probability
      for each row of matrix
      if (flags < pro)
7       keep current row
      break
    end
  end
8 CO:
9 Repeat m/(λ_g ∗ 0.8 + 5) times
10  calculate the P_c by equation (12)
11  random a seed
    if seed < P_c
12    cross operation
    end
  end
13 MO:
14 Repeat m times traversing the row
15  calculated the P_m of the current row
16  random a seed
17  if seed < P_m
18    Mutation operation
    end
  end
```

4.3 An MO approach based on mutation intensity control

In the MO stage, each offloading queue combination will calculate the probability P_m according to equation (13) to obtain a new offloading task queue. In other words, the offloading queues with poor fitness values are more likely to change into new offloading queues. In the MO of Section 4, M-COGA randomly selects the two points of the queue to exchange the task, when an offloading task queue is determined to mutate. The values in the individual queue, which represent the arrangement offloading tasks. And the offloading tasks at the back of the queue can not be uploaded due to resource constraints. New offloading queues, which are generated by mutations in two near positions, are not conducive to obtain better offloading task sequences. Therefore, we design a strategy during the mutation operation to generate a new task queue. We define the mutation distance D_i, which represents the distance between two swapped positions. Besides, the offloading task queue, which is represented as a row, is divided into three ranges, which is shown as Fig. 2.

Then, we can describe our strategy as follows:

It is assumed that the row i is determined to execute MO with probability P_m , the two interchanged tasks x_i, x_j will be selected at the different ranges. The selection strategy is depended on the equation (15).

$$\varpi(i) = \frac{\alpha'(i) - \zeta}{\delta' - \zeta}, \tag{15}$$

where the $\alpha'(i)$ represents the fitness value of row i, the ζ and δ' mean the minimum and maximum fitness value of In_c, respectively. If the $\varpi(i) \in (0, 1/3]$, the two tasks will be randomly selected in the first range and the third range. If the $\varpi(i) \in (1/3, 2/3]$, the two tasks will be randomly selected in the first range and the second range. If the $\varpi(i) \in (2/3, 1]$, both of two tasks will be randomly selected in the first range. In this way, an offloading task queue with a lower fitness value will have a higher D_i. It means that MO will facilitate the generation task queue with better fitness value. To observe the convergence performance with the new strategy, we give the convergence times as Table 2 under different device densities according to the parameter setting in Section 5.

The algorithm is set to exit the calculation when the fitness is unchanged for 250 times. We found that the strategy effectively reduced the number of iterations except in the 30 devices scenario.

4.4 The dynamic task offloading strategy with M-COGA

As it is shown in equation (6), the computation task offloading depends on the bandwidth $\bar{B}$ of a BS. However, in the actual task offloading scenario, the transmission time is different due to various size of tasks. In the case of

Table 2 The iteration count at different node densities

Algorithm \ Devices	30	40	50	60
M-COGA with random MO	431	541	628	892
M-COGA with optimized MO	566	528	537	656

constraint value $\bar{B}$, M-COGA will be restarted after the offloaded all pending tasks are complete. Therefore, M-COGA periodically checks the available channel capability $\bar{C}_a^{(\mathcal{B})}$, which can be described as equation (16)

$$\bar{C}_a^{(\mathcal{B})} = C_{\bar{s}}^{(\mathcal{B})} - \sum_{i=1}^{n} Tr(i), \tag{16}$$

where $Tr(i)$ is the transmission rate of task i. And n represents the total number of tasks being transmitting. Then, the rest of the pending tasks will process by M-COGA based on $\bar{C}_a^{(\mathcal{B})}$.

5 Evaluation results

We implemented the proposed scenario and the M-COGA algorithm. Then, several simulation results are presented for evaluating the performance of the M-COGA strategy. We have considered a variety of scenarios, mainly observed in the performance of three aspects: The M-COGA in single BS scenario, The M-COGA in multiple BSs scenario with random BS selection and with BS selection strategy. In the multiple BSs scenario, we also consider the uniform and non-uniform distribution of terminals in each BSs' coverage area.

The parameters of different types of base stations are also different. For instance, Huawei BBU5900 supports up to 400 users in a single cell. A single cell covers an area of about 400 meters or more, which depends on the building block and the terrain. And the coverage can be flexibly adjusted for users [21]. However, not all of the terminals covered by a BS will have the requirement of task offloading, and the distribution of nodes is uneven in the city areas. Therefore, the offloading terminals will be less than 400 devices in a cell. Generally speaking, the uploading task size is related to the application of offloading tasks in practice. There are up to 120 mobile devices in a 1000m*1000m square area. Three BSs are distributed in this scenario, and the cover radius of each BS is 400m. The channel gain is not considered in this scenario due to the principle of the algorithm. The BS bandwidth is generally limited by radio management, and only part of the BS's bandwidth is allowed for the purpose of task uploading. Therefore, the parameter of channel bandwidth is assumed as $C_{\bar{U}} = 5MHz$ [22]. The transmission power $S_i = 100mw$ and the noise $N_0 = -100dbm$ [10]. In this scenario, each device owns a task for offloading computing requirements, the $\mathcal{J}_i = 8.9 * 10^{-12}$J/cycle. The amount of offloading data size is randomly generated in the range of $(0.175 * 10^5, 0.5 * 10^5)$ bit/s. The coefficient λ_t and λ_e are set as 0.3, 0.7, separately. Since the task data is randomly generated, then, the M-COGA runs to select offloading tasks.

To verify the validity of the method, each experiment is carried out five times. The performance should be tracked at five aspects in different task density: the convergence status, the remaining capacity of the channel, the offloading tasks, offloading cost and the processing time with the proposed algorithms.

The (a) of Fig. 3 displays the algorithm convergence for a single BS scenario in different task density. It shows that the fitness value can effective convergence with M-COGA. And it works under smaller the number of tasks, the faster the convergence. As it is shown in (b) of Fig. 3, the fitness value also convergence under multiple BSs scenario. From Fig. 4, we observed the remaining bandwidth for different number of offloading tasks, there is more capacity left due to the small data offloading requirements. Besides, when ultra tasks require task offloading to BS, the remaining capacity of channel maintains at a low level. It shows the full utilization of the transmission ratio with M-COGA. In the multiple BSs scenario, there are multiple base stations in the

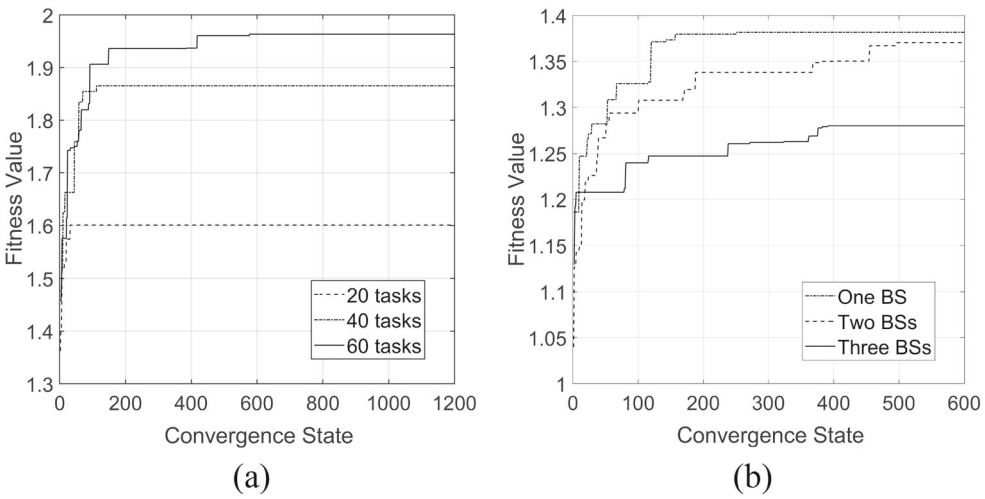


Fig. 3 The convergency of M-COGA in variety of scenarios

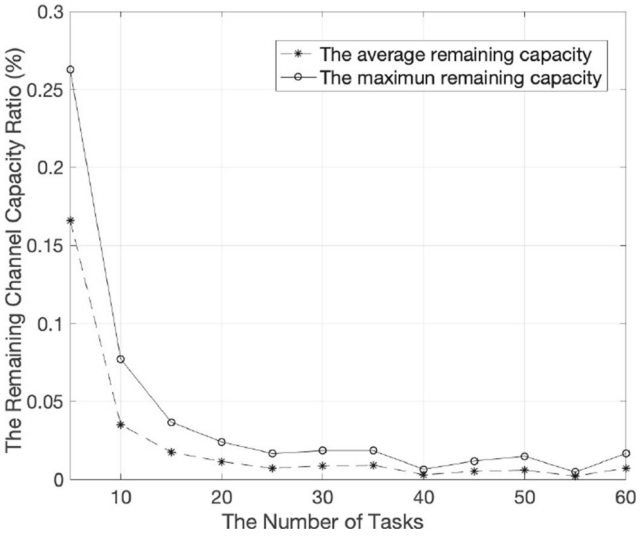


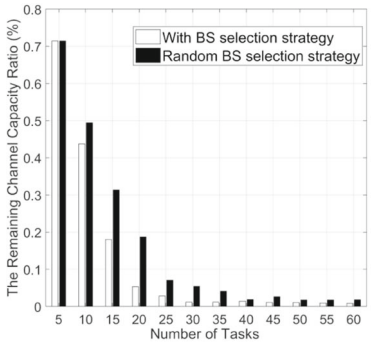
Fig. 4 The channel capability with M-COGA under single BS scenario

node communication area, so the requirement of computing task offloading is more complicated. The distribution of devices in demand for task offloading may be uneven, then we consider both uniform and non-uniform nodes distribution to observe the remaining capability of BSs.

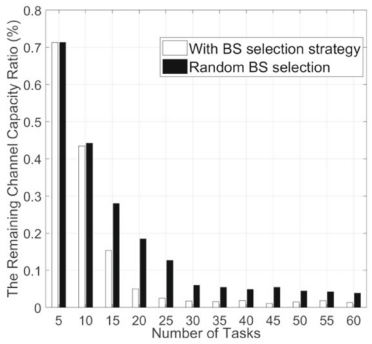
Figure 5(a) and (b) show that the M-COGA with BS selection strategy can work better than without the BS selection strategy under both non-uniform and uniform distribution of devices in 3 BSs scenario. From the (c) and (d) of Fig. 5, we found that the M-COGA with BS selection strategy can achieve the load balancing of the BSs as well, even if the nodes are not evenly distributed. In intensive task scenarios, the remaining space is very small. Therefore, it shows the effectiveness of residual capability control.

Both the terminal cost and the number of offloading tasks are the optimization targets of the M-COGA, then we respectively observed its performance in various aspects. It includes Uniform Distribution (UD) with BS Selection Strategy (SS), uniform distribution without selection strategy, Non-uniform Distribution (ND) with BS selection strategy and non-uniform distribution without BS selection strategy. From the results, with the increase of the density of pending tasks, the number of offloading tasks and the total cost are increasing. In the case of limited channel capacity, a relative balance is achieved. And the results are stable in the relatively intensive task scenes, which indicates that it near the best-optimized result in the current bandwidth situation. In general, M-COGA with the BS selection strategy is better than random BS selection strategy under the ultra density of pending tasks, therefore, the BS selection strategy is effective with M-COGA.

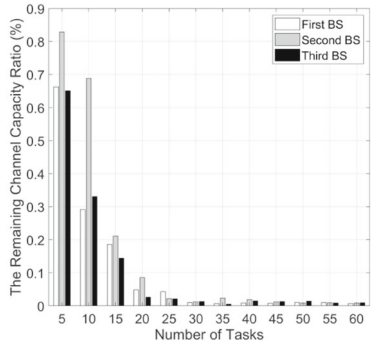
When M-COGA selects offloading tasks under fixed channel capacity, the selected task has a different transmission time. Therefore, we chose a dynamic task offloading



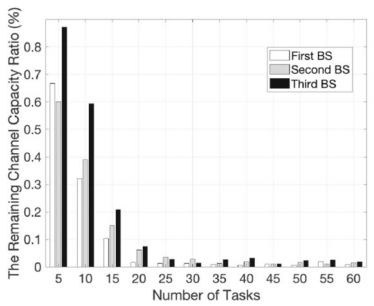
(a) Average remaining capability ratio under uniform distribution of devices in the three BSs scenarios



(b) Average remaining capability ratio under non-uniform distribution of devices in the three BSs scenarios



(c) Each BS remaining capability ratio under uniform distribution of devices

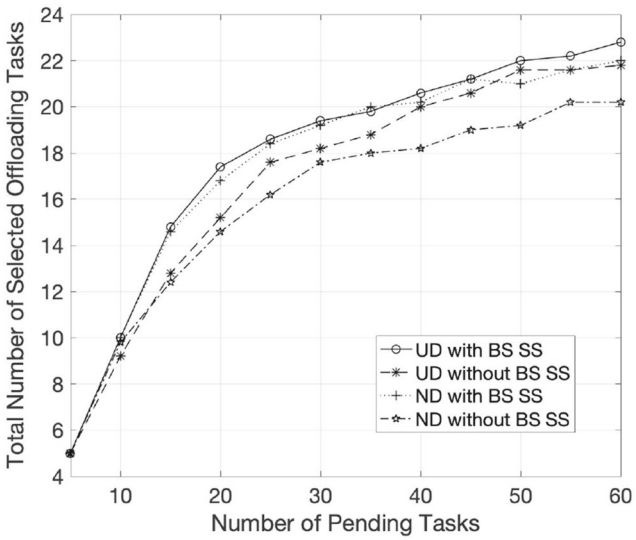


(d) Each BS remaining capability ratio under non-uniform distribution of devices

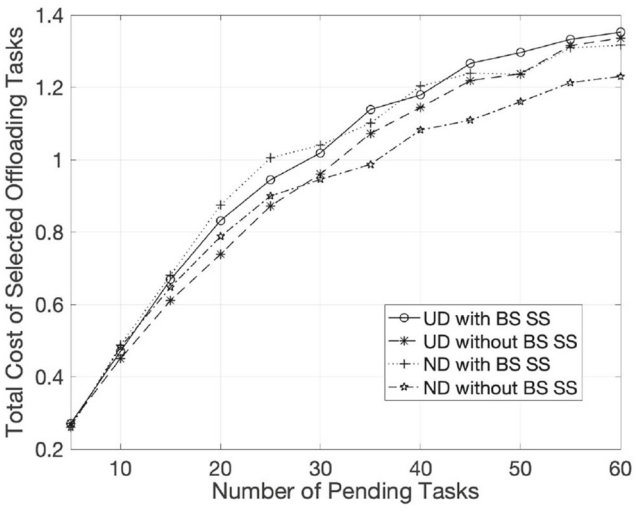
Fig. 5 The channel capability with M-COGA under multiple BS scenarios

strategy. If a task finished offloading, then it will release the occupied channel capacity in ultra density of pending task. We established the scenario of Fixed Channel Capacity (FCC) with or without BS selection strategy and Dynamic Offloading (DO) policy with or without BS selection strategy.

Figure 6 shows the performance of the M-COGA in four scenarios. We can see that the BS selection strategy can get better performance to some extent. Since the allocated bandwidth may be different by task offloading, we also observe the M-COGA effect under different bandwidth conditions. We have compared the M-COGA with the random and the Greedy offloading algorithm.

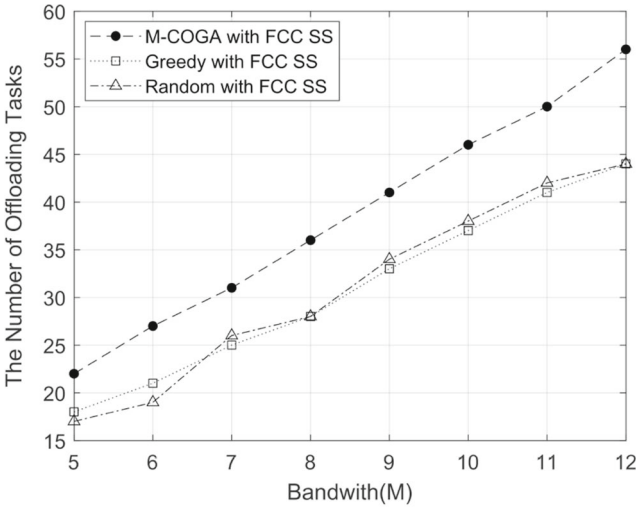


(a) The number of offloading tasks

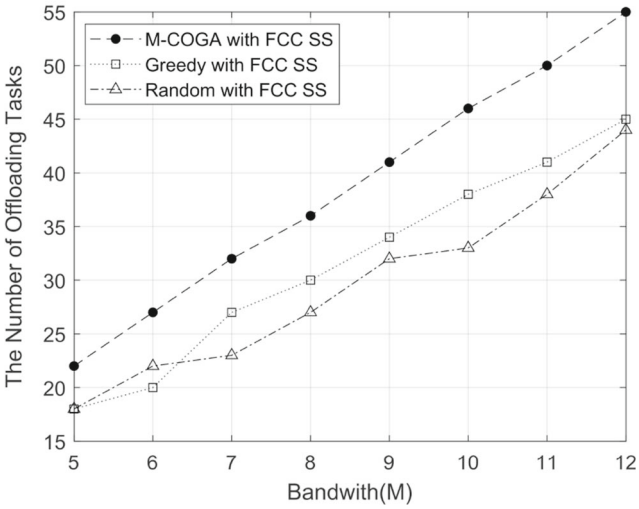


(b) The cost of offloading tasks

Fig. 6 The total number and the cost of offloading tasks in four scenarios



(a) Uniform distribution of devices



(b) Non-uniform distribution of devices

Fig. 7 The number of task offloading tasks of the three algorithms under various bandwidths

Figure 7 indicates that the performance of M-COGA is still better than random and Greedy under different bandwidth scenarios.

From Fig. 8, the dynamic offloading strategy with M-COGA takes minimal time to process the computing offloading tasks for the same requirements. In the fixed channel capacity scenario, the algorithm with BS selection is also acceptable. In addition, the BS selection strategy still keeps relative low processing time under ultra task density circumstance. Therefore, dynamic unloading strategy and BS selection strategy promote the time efficiency of M-COGA in computing task offloading.

We also observed the performance of M-COGA on the number and cost of computing offloading tasks in a dynamic offloading scenario. From Fig. 9(a), The M-COGA with BS

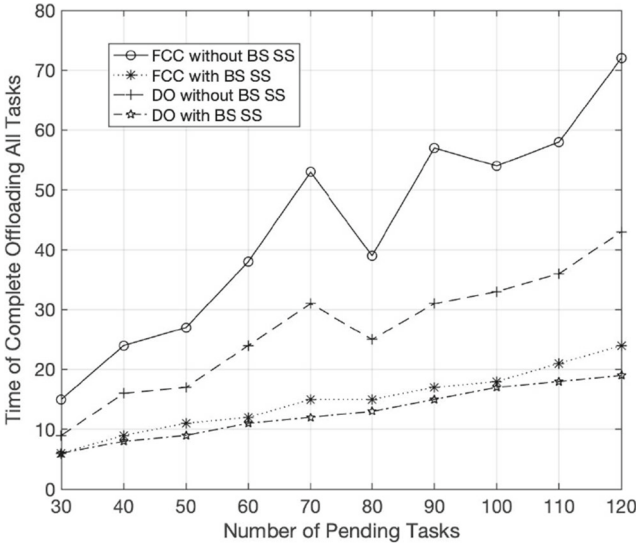
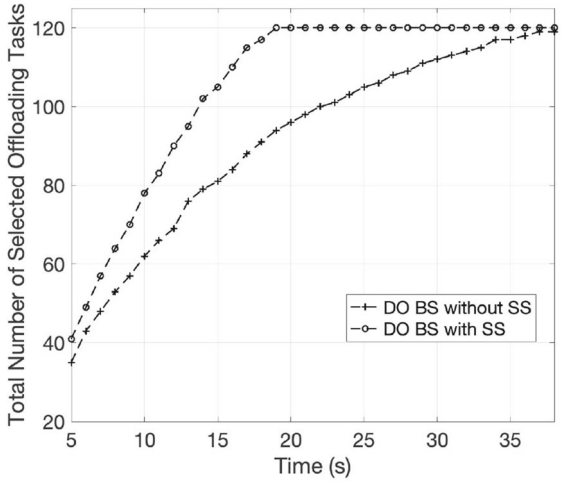
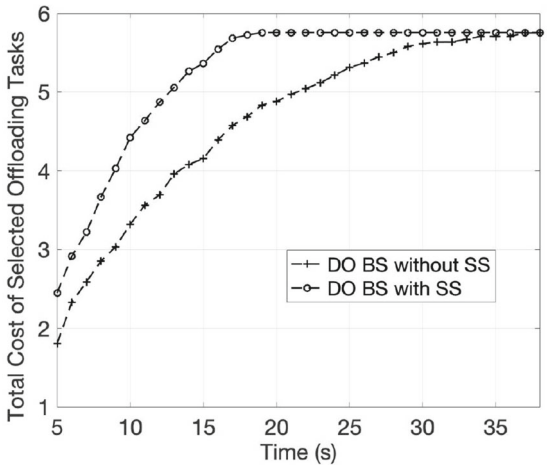


Fig. 8 The cost of offloading tasks in different task density



(a) The number of offloading tasks over time



(b) The cost of offloading tasks over time

Fig. 9 The total number and the cost of offloading tasks over time in dynamic offloading scenarios

selection strategy is much better than M-COGA without the BS selection strategy. The same as the cost of offloading in Fig. (9) (b), M-COGA can get to a better solution as quickly as possible.

Therefore, from the above comparison results, M-COGA can effectively reduce the terminal computation load. And it also achieves the design purpose of the most offloading tasks under the limited channel capacity. The BS selection strategy comes out superior performance in both static and dynamic task offloading schemes under multiple BSs scenario.

6 Related work

In the research areas of computation offloading, the proposed approaches aim to energy, latency, and joint consideration, and most of them consider the offloading processing between the terminal and the cloud [23–25]. Zhao et al. [26] discussed the task scheduling based on the consideration of computing limitation in the edge cloud. The task offloading strategy aims to reduce task latency by coordinating the heterogeneous cloud model. Li et al. [27] aim to optimize the formulated cost model in multi-users scenarios by Deep Reinforcement Learning (DRL) method [28]. The results also achieve the proposed design purpose. The resource allocation approach in MEC is one of the key questions, and there are more studies for whole network performance progress. Changsheng et al. [18] concentrate on energy consumption and computation latency at multi-user scenarios. The proposed algorithm was designed based on priority policy to reduce the search cost. In similar task offloading scenarios, Xu et al. [10] formulate the several computation decision making among devices as a game in MEC. Min et al. [29] proposed an offloading schedule to optimize the delay and battery life of users. This approach can reduce the task duration and energy cost in the software-defined ultra-dense network. This formulated problem aims to allocated resources between cloud and edge cloud. Bi et al. [14] also consider the computing mode which includes offloading and local computing. And the transmission time is also considered in the enumeration-based optimal method. Zhiguo et al. [30] proposed a hybrid NOMA strategy which can permit partial task offloading by using a time slot. It jointly considers the power and time allocation in NOMA scenarios. These existing studies looked at both latency, energy efficiency, or physical layer. However, in the actual application scenarios, the number of

tasks and the above factors will affect the MEC computation experience. In addition†, under the multi-BS coverage, there are more flexible selection strategies to improve the offloading performance.

From some of the recent researches above, the computation offloading is a complex problem. Each method focuses on the different aspects of MEC systems. The optimized problem ignores the task numbers and decreases the maximum cost of terminal devices which are covered in the communication area of BS. Then we proposed the M-COGA to optimize the number of offloading tasks and computing cost in MEC.

7 Conclusion

With the rapid expansion of mobile applications or smart vehicle networks, more and more computing tasks will appear in the future [31]. It is reasonably prophesied that the task computation offloading will become more and more important in IoT, due to the satisfaction of a task offloading requirement can probably satisfy the needs of an intelligent application. Therefore, we proposed an M-COGA computation offloading approach based on the genetic algorithm, which decides the task offloading and adjusts the mobile terminal cost for energy efficiency, the number of offloading tasks and computing resources under limited channel capacity. Then, we not only consider the intensive tasks in one BS scenario but also proposed the BS selection strategy under multiple BSs scenario. After plenty of simulations, several results indicated that the proposed algorithm and BS selection strategy can effectively determine the offloading tasks. Besides, we also have finished the dynamic task selection scheme, which can improve the time efficiency of offloading processing. The method has the flexibility for optimized targets, it still has some aspects that have not yet been discussed. For some instances, we have not considered the CPU computing power of BSs, or some tasks with a large amount of data can not be treated fairly. And the traces of the real world should be considered to evaluate algorithms. The valuable problems deserve further discussions in the future work.

Acknowledgment This research was supported by China Scholarship Council (CSC), Fund of Applied Basic Research Programs of Science and Technology Department (No.2018JY0290), F. Wang and J. Liu's research is supported by an NSERC Discovery Grant, The authors also thank for the supporting of SINOPEC Key Laboratory of Geophysics and Graphic & image Collaborative Innovation Center of CUIT.

Compliance with Ethical Standards

Conflict of interests The authors declare that there is no conflict of interests regarding the publication of this paper.

References

1. Teli SR, Zvanovec S, Ghassemlooy Z (2018) Optical internet of things within 5g: Applications and challenges. In: 2018 IEEE International Conference on Internet of Things and Intelligence System (IOTAIS), pp 40–45. IEEE
2. Liu X, Cao J, Yang Y, Qu W, Zhao X, Li K, Yao D (2019) Fast rfid sensory data collection: Trade-off between computation and communication costs. IEEE/ACM Trans Networking 27(99):1179–1191
3. Liu X, Xie X, Wang S, Liu J, Yao D, Cao J (2019) Efficient range queries for large-scale sensor-augmented rfid systems. In: EEE/ACM Transactions on Networking (TON), p. In press
4. Wang F, Wang F, Ma X, Liu J (2019) Demystifying the crowd intelligence in last mile parcel delivery for smart cities. IEEE Netw 33(2):23–29
5. Mao Y, You C, Zhang J, Huang K, Letaief KB (2017) A survey on mobile edge computing: The communication perspective. IEEE Communications Surveys & Tutorials 19(4):2322–2358
6. Zhao X, Zhao L, Liang K (2016) An energy consumption oriented offloading algorithm for fog computing. In: International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, pp 293–301. Springer
7. Mao Y, Zhang J, Song SH, Letaief KB (2017) Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems. IEEE Trans Wirel Commun 16(9):5994–6009
8. Satyanarayanan M (2017) The emergence of edge computing. Computer 50(1):30–39
9. Li Z, Zhu Q (2020) Genetic algorithm-based optimization of offloading and resource allocation in mobile-edge computing. Information-an International Interdisciplinary Journal 11(2):83
10. Chen X, Jiao L, Li W, Fu X (2015) Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Trans Networking 24(5):2795–2808
11. Hao Y, Ni Q, Li H, Hou S (2019) Energy-efficient multi-user mobile-edge computation offloading in massive mimo enabled hetnets. In: ICC 2019-2019 IEEE International Conference on Communications (ICC), pp 1–6. IEEE
12. Tran TX, Pompili D (2019) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. IEEE Trans Veh Technol 68(1):856–868
13. Guo S, Xiao B, Yang Y, Yang Y (2016) Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing. In: IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, pp 1–9. IEEE
14. Bi S, Zhang YJ (2018) Computation rate maximization for wireless powered mobile-edge computing with binary computation offloading. IEEE Trans Wirel Commun 17(6):4177–4190
15. Di B, Bayat S, Song L, Li Y, Han Z (2016) Joint user pairing, subchannel, and power allocation in full-duplex multi-user ofdma networks. IEEE Trans Wirel Commun 15(12):8260–8272
16. Bockelmann C, Pratas N, Nikopour H, Au K, Svensson T, Stefanovic C, Popovski P, Dekorsy A (2016) Massive machine-type communications in 5g: Physical and mac-layer solutions. IEEE Commun Mag 54(9):59–65
17. Du Y, Dong B, Chen Z, Fang J, Yang L (2016) Shuffled multiuser detection schemes for uplink sparse code multiple access systems. IEEE Commun Lett 20(6):1231–1234
18. You C, Huang K, Chae H, Kim B-H (2016) Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans Wirel Commun 16(3):1397–1411
19. Pisinger D (2005) Where are the hard knapsack problems? Computers & Operations Research 32(9):2271–2284

20. Chen R, Liang C-Y, Hong W-C, Gu D-X (2015) Forecasting holiday daily tourist flow based on seasonal support vector regression with adaptive genetic algorithm. Appl Soft Comput 26:435–443
21. Official H Huawei launches full range of 5g end-to-end product solutions. [EB/OL]. https://www.huawei.com/en/press-events/news/2018/2/Huawei-Launches-Full-Range-of-5G-End-to-End-Product-Solutions. Accessed Feb 26, 2018
22. Chen X (2015) Decentralized computation offloading game for mobile cloud computing. Parallel and Distributed Systems IEEE Transactions on 26(4):974–983
23. Kwak J, Kim Y, Lee J, Chong S (2015) Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems. IEEE Journal on Selected Areas in Communications 33(12):2510–2523
24. Huang D, Wang P, Niyato D (2012) A dynamic offloading algorithm for mobile computing. IEEE Trans Wirel Commun 11(6):1991–1995
25. Chen S, Wang Y, Pedram M (2013) A semi-markovian decision process based control method for offloading tasks from mobile devices to the cloud. In: 2013 IEEE Global Communications Conference (GLOBECOM), pp 2885–2890. IEEE
26. Zhao T, Zhou S, Guo X, Niu Z (2017) Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In: 2017 IEEE International Conference on Communications (ICC), pp 1–7. IEEE
27. Li J, Gao H, Lv T, Lu Y (2018) Deep reinforcement learning based computation offloading and resource allocation for mec. In: 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp 1–6. IEEE
28. Wang F, Zhang C, Liu J, Zhu Y, Pang H, Sun L et al (2019) Intelligent edge-assisted crowdcast with deep reinforcement learning for personalized qoe. In: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp 910–918. IEEE
29. Chen M, Hao Y (2018) Task offloading for mobile edge computing in software defined ultra-dense network. IEEE Journal on Selected Areas in Communications 36(3):587–597
30. Ding Z, Xu J, Dobre OA, Poor V (2019) Joint power and time allocation for noma-mec offloading. IEEE Trans Veh Technol 68(6):6207–6211
31. Wang F, Zhu Y, Wang F, Liu J, Ma X, Fan X (2019) Car4pac: Last mile parcel delivery through intelligent car trip sharing. IEEE Trans Intell Transp Syst PP(99):1–15. https://doi.org/10.1109/TITS.2019.2944134