Optimization of Cooperative Offloading Model with Cost Consideration in Mobile Edge Computing

Bin Xu (✉ xubin2013@njupt.edu.cn)

Nanjing University of Posts and Telecommunications <https://orcid.org/0000-0001-7837-8554>

Tao Deng

Nanjing University of Posts and Telecommunications

Yichuan Liu

Nanjing University of Posts and Telecommunications

Yunkai Zhao

Nanjing University of Posts and Telecommunications

Zipeng Xu

Nanjing University of Posts and Telecommunications

Jin Qi

Nanjing University of Posts and Telecommunications

Sitao Wang

Syracuse University

Dan Liu

Nanjing University of Posts and Telecommunications

Research Article

Keywords: mobile edge computing, computation offloading, adaptive neighborhood search, simulated annealing algorithm, multiuser collaborative model

Posted Date: January 10th, 2022

DOI: <https://doi.org/10.21203/rs.3.rs-165316/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Optimization of Cooperative Offloading Model with Cost Consideration in Mobile Edge Computing

Bin Xu^{1,2,3} · Tao Deng¹ · Yichuan Liu¹ · Yunkai Zhao¹ · Zipeng Xu¹ · Jin Qi^{1,2} · Sitao Wang⁴ · Dan Liu¹

Received: date / Accepted: date

Abstract The combination of idle computing resources in mobile devices and the computing capacity of mobile edge servers enables all available devices in an edge network to complete all computing tasks in coordination to effectively improve the computing capacity of the edge network. This is a research hotspot for 5G technology applications. Previous research has focused on the minimum energy consumption and/or delay to determine the formulation of the computational offloading strategy but neglected the cost required for the computation of collaborative devices (mobile devices, mobile edge servers, etc.); therefore, we proposed a cost-based collaborative computation offloading model. In this model, when a task requests these devices' assistance in computing, it needs to pay the corresponding calculation cost; and on this basis, the task is offloaded and computed. In addition, for the model, we propose an adaptive neighborhood search based on simulated annealing algorithm (ANSSA) to jointly optimize the offloading decision and resource allocation with the goal of minimizing the sum of both the energy consumption and calculation cost. The adaptive mechanism enables different operators to update the probability of selection according to historical experience and environmental

perception, which makes the individual evolution have certain autonomy. A large number of experiments conducted on different scales of mobile user instances show that the ANSSA can obtain satisfactory time performance with guaranteed solution quality. The experimental results demonstrate the superiority of the mobile edge computing (MEC) offloading system. It is of great significance to strike a balance between maintaining the life cycle of smart mobile devices and breaking the performance bottleneck of MEC servers.

Keywords mobile edge computing · computation offloading · adaptive neighborhood search · simulated annealing algorithm · multiuser collaborative model

1 Introduction

Exponentially increasing data volumes and diverse service scenarios are increasing the demands on mobile device performance. Although the central processing unit (CPU) of a mobile device has undergone great development, sometimes it still cannot meet the demand for processing a large number of computing tasks in a short time. In addition, the limited battery energy and storage space have also become important factors that restrict mobile devices from processing large-scale data computing applications in a timely manner and affect the quality of the user experience (Abbas et al., 2017).

By offloading the massive amount of data required on a mobile device to the cloud, mobile users can use more complex and computation-intensive programs while extending the battery life of the device itself. However, this method transmits a large amount of data to a cloud server, which causes a large delay. In addition, mobile terminals need to consume more energy, which increases the network load and affects the service quality (Wei

✉ Bin Xu
E-mail: xubin2013@njupt.edu.cn

Jin Qi
E-mail: qijin@njupt.edu.cn

¹ Nanjing University of Posts and Telecommunications, Nanjing, 210003, China

² NanJing Pharmaceutical Co., Ltd., Nanjing, 210012, China

³ Jiangsu Key Laboratory of Data Science and Smart Software, Jinling Institute of Technology, Nanjing, 211169, China

⁴ Syracuse University, State of New York, 13244, United States

et al., 2021). In order to get rid of this limitation, the Mobile Edge Computing (MEC) mode came into being. To improve the user experience, the engineers provide users with good IT environments and powerful computing power by erecting edge computing servers closer to users (Chen and Hao, 2018). Due to the powerful computing power of edge devices, we can embed some artificial intelligence algorithms into edge devices to meet the requirements of different scenes (Gumaei et al., 2020; Liu et al., 2020). However, there are some disadvantages to an MEC server. When increasingly more tasks are relocated to an MEC server, serious interference will occur between the devices, which increases the delay. And too many tasks may overload an MEC server. In addition, we need to consider the security of edge computing, which is the difficulty in the research (Qian et al., 2020).

Some scholars have proposed to integrate collaborative computing technology into a mobile edge computing offloading system to solve the above problems. There are numerous idle mobile devices which play the role of virtual MEC servers that share computing resources with other devices in the network, in the existing multiuser collaboration model (Pu et al., 2016; Ti and Le, 2017; Cao et al., 2018), the offloading decision is mainly determined using the delay or energy consumption as the optimization goal. However, in fact, after adding synergy technology, it is not reasonable to only use these indicators as optimization targets. Since users have no commitments for cooperation, one important aspect of co-computing research is to design schemes for incentivizing them for sharing computation resources. The multiuser collaborative MEC offloading model proposed in this paper considers multiuser interference and delay-sensitive task acceptance and proposes an index based on the collaborative technology's calculation cost to optimize the target selection, not only energy consumption. Various types of collaborative devices will require different calculation costs due to factors such as status and environment. When a task requests devices' assistance in computing, it needs to pay the corresponding calculation cost. The index of collaborative calculation cost can indirectly quantify the cost required to motivate users to share resources for improving the performance of co-computing. This makes the model more reasonable and more practical.

The solution of the mobile edge computing offloading decision scheme is a complex non-deterministic polynomial (NP) hard problem (Guo et al., 2017). The self-adaptive mechanism plays an important role in solving practical problems (Qasim and Kazmi, 2019). By using it appropriately, the solution has good global and local search ability when dealing with NP hard problems

(Xue et al., 2020). We propose an adaptive neighborhood search based on simulated annealing algorithm (ANSSA). The ANSSA adopts an adaptive mechanism to update the probability of different operators being selected according to the historical evolution experience to realize the adaptive evolution of the algorithm and effectively improve the solution performance.

The main contributions of this paper are as follows:

- Based on the collaborative computing, the calculation cost is incorporated for the first time, and it is used as the optimization goal to further improve the rationality of the model. It also indirectly quantifies the cost of motivating users to share resources and further improves the performance of collaborative computing.
- We propose a novel algorithm based on the simulated annealing adaptive neighborhood search algorithm to quickly solve the multiuser unloading problem. It adopts an adaptive mechanism to update the probability that different operators are selected based on historical experience and context perception, which allows individual evolution to have certain autonomy. It strives for individuals to choose the optimal strategy in each iteration.
- A large number of experiments on different numbers of mobile users show that the ANSSA can achieve satisfactory performance in all instances compared to the two other algorithms. The superiority of the MEC offloading system has been verified.

2 Related work

2.1 Solving the offloading decision and resource allocation problem

In recent years, as 5G technology develops, MEC has become a focus of research effort. Mao et al. (2016) propose a mobile-edge computing model for a single-user-single MEC server with the shortest task execution time as the optimization goal. Another study (Wei et al., 2018) studies the problem of the joint allocation of computing and communication resources, aiming at achieving energy savings in the multiuser-single MEC server scenario. Other authors (Zhang et al., 2016) consider the interference between multiuser equipment in a multiuser mobile edge computing offloading system, and minimize the system energy consumption under the delay constraint by optimizing the offloading decision. However, none of the above studies consider that the offloading decision and the allocation of computing resources are interdependent. The allocation of computing and communication resources depends on the for-

1 mulation of the offloading decision, and the quality of
 2 the offloading decision changes with the distribution of
 3 resources. Therefore, it is necessary to jointly optimize
 4 both of them.

5 Some scholars have considered this issue. Huang
 6 et al. (2019) consider the multitask interference situa-
 7 tion, the offloading decision and resource allocation are
 8 jointly optimized to achieve the minimum system en-
 9 ergy consumption. Tran and Pompili (2018) design a
 10 multiunit, multiservice offloading system by jointly op-
 11 timizing the offloading decision and resource allocation.
 12 Stackelberg game theory is used to find the optimal de-
 13 cisions and optimal resource allocation for multiuser-
 14 single MEC server scenarios (Liu and Liu, 2017). How-
 15 ever, although the methods in the above studies were
 16 able to obtain better solutions in the solution process,
 17 the solution times are too long, and a user may be out-
 18 side the effective range by the time a good solution is
 19 available. Furthermore, the above algorithms easily fall
 20 into local optima.

21 The characteristic of the simulated annealing (SA)
 22 is that the inferior solution can be accepted with a cer-
 23 tain probability in order to escape the local optimum
 24 to a certain extent. WANG and YU (2011) research a
 25 blind detection algorithm based on simulated anneal-
 26 ing chaos particle swarm optimization, and find that
 27 the improved algorithm achieved a good improvement
 28 in the global convergence. Zheng et al. (2013) com-
 29 bine Particle Swarm Optimization with Gaussian kernel
 30 function (GPSO) and the SA algorithm to form a new
 31 hybrid algorithm. GPSO is used in the early stage, and
 32 the SA algorithm is used to escape the local optimum to
 33 find the global optimum when GPSO is stagnant. The
 34 above work shows that the combination of the SA algo-
 35 rithm and other algorithms can effectively prevent the
 36 premature convergence of the algorithm. We combine
 37 the simulated annealing algorithm with the adaptive
 38 neighborhood search algorithm and add many differ-
 39 ent evolution operators, effectively improving the search
 40 and exploration capabilities of the algorithm. In addi-
 41 tion, we adopt a two-layer optimization method (Huang
 42 et al., 2019) to jointly optimize the offloading decision
 43 and resource allocation to greatly improve the quality
 44 of the results.

51 2.2 Combination of collaborative computing and 52 mobile edge computing

53 In recent years, some scholars have proposed combin-
 54 ing collaborative computing technology with MEC to
 55 enhance its abilities. A device-to-device (D2D) atom-
 56 ization framework is proposed to achieve joint comput-
 57 ing and network resource sharing among mobile users

(Pu et al., 2016). Reference (Ti and Le, 2017) proposes
 a general framework for transferring device computing
 tasks to edge clouds and/or moving peers. Cao et al.
 (2018) study a three-node MEC offload system consist-
 ing of a user node, a secondary node and an access point
 node connected to a MEC server, which divides the
 user's computing tasks into three parts: local comput-
 ing, assistant offload and AP offload. Based on these,
 we consider the delay-sensitive task and the channel
 interference between multiple users, making the model
 more complicated. In addition, we propose a new in-
 dicator - the calculation cost. When a device offloads
 its task to other devices for execution, it needs to pay
 a certain price to the corresponding device as compen-
 sation for using the device's resources. The index indi-
 cally quantifies the cost of motivating users to share
 resources and further improves the performance of col-
 laborative computing. This paper jointly optimizes the
 energy consumption and calculation cost and minimizes
 the sum of their values to construct a more reasonable
 and practical model.

3 System model

As shown in Fig. 1, we consider a multiuser collabo-
 ration model consisting of a base station with a MEC
 server denoted as S and a set of n mobile device users
 denoted as $N = \{1, 2, 3 \dots n\}$. We assume that each
 mobile user will only post one computing task, and de-
 note the task of mobile user i as $A_i (i \in N)$, which can
 be defined by a 5-tuple: $A_i = (C_i, D_i, B_i, T_i^{\max}, Co_i)$.
 Herein, C_i denotes the number of CPU cycles required
 to complete task A_i ; D_i and B_i denote the sizes of the
 input data and results, respectively; $T_i^{\max}$ represents
 the maximum delay that task A_i can tolerate; and Co_i
 represents the calculation cost of task A_i when running
 on the device.

We model the entire network as a random directed
 graph denoted as $\{V, E\}$. $V = N \cup S$ represents the
 collection of mobile devices and MEC servers in the en-
 tire network. $E = \{(i, j) \mid e_{ij} = 1, \forall i, j \in V\}$ indicates
 the connection status in all nodes. Herein, $e_{ij} = 1$ rep-
 resents node i initiating a connection to node j and rep-
 resents an unloading decision. We assume that the entire
 scenario is static, that is, it does not change regardless
 of the impact of the mobile device during the move. In
 addition, r_{ij} indicates that task A_i selects node j to
 perform the required computing resource allocation.

Fig. 1: A multiuser collaboration model consisting of a base station equipped with a MEC server and n mobile device users

3.1 Energy consumption calculation

When given task A_i selects node j to execute the required computing resource r_{ij} , the calculation time T_{ij}^c that task A_i needs to consume is represented by Equation (1). In addition, $r_{ij} \leq f_j$ and f_j is the maximum computing power of node j itself.

$$T_{ij}^c = \frac{C_i}{r_{ij}} \quad (1)$$

We only consider the computational power consumption on mobile devices because mobile devices usually rely on batteries to keep working, and cannot last for a long time. Computation-intensive tasks quickly consume power for mobile devices. So the computational task energy consumption ε_{ij}^c can be expressed as equation (2).

$$\varepsilon_{ij}^c = \begin{cases} e_{ijk}(r_{ij})^2 C_i, & i \in N, j \in V \setminus \{S\} \\ 0, & i \in N, j = S \end{cases} \quad (2)$$

The computing task energy consumption is related to the hardware circuit structure of the mobile device itself, which is represented by a constant coefficient k .

In the proposed model, the MEC server can accept multiple tasks. Considering that multiple tasks in the same mode will share a channel and cause interference, according to Shannon's formula, the maximum transmission rate R_{ij} can be obtained when task A_i selects node j . However, it is worth noting that when selecting the local device mode, there is no need to transfer data. This is shown in equation (3).

$$R_{ij} = \begin{cases} \infty, & i \in N, j \in V \text{ and } j = i \\ B^* \log_2 \left(1 + \frac{p_i^{tr} H_{ij}}{\delta^2 + \sum_{i=1}^n e_{ij} p_i^{tr} H_{ij} - p_i^{tr} H_{ij}} \right), & \text{otherwise} \end{cases} \quad (3)$$

where B is the bandwidth, H_{ij} is the channel gain when node i communicates with node j , p_i^{tr} is the power when node i sends the task data, and δ^2 is the noise power. Similarly, after the task data is transmitted and calculated, the result data needs to be returned to device i . We assume that the transmission rate of the outgoing data is consistent with the transmission rate of the returned result data, that is, that $R_{ij} = R_{ji}$.

Then, we can obtain the transmission time T_{ij}^{tr} of the task data, as shown in equation (4).

$$T_{ij}^{tr} = \frac{D_i}{R_{ij}} + \frac{B_i}{R_{ij}} \quad (4)$$

Obviously, when $j = i$, that is, when the mode is local mode, there is no need to transmit data. Hence, $T_{ij}^{tr} = 0$.

For the energy consumption ε_{ij}^{tr} generated during the transmission process, if the calculation task A_i is performed in the cooperation mode, not only does node i need to send the task data to receive the data result, but node j must also perform the calculation task when receiving the task data and returning the data result. To determine the energy consumption, this energy consumption is also included in this task. p_i^{re} is the power when node i receives the task data. However, when the collaborative device is a MEC server, this part of the energy consumption is borne by the MEC server, and the transmission energy consumption is not included. Therefore, the formula for calculating ε_{ij}^{tr} is as shown in equation (5).

$$\varepsilon_{ij}^{tr} = \begin{cases} e_{ij} \left(\frac{p_i^{tr} D_i}{R_{ij}} + \frac{p_j^{re} D_i}{R_{ij}} \right) + e_{ji} \left(\frac{p_j^{tr} B_i}{R_{ji}} + \frac{p_i^{re} B_i}{R_{ji}} \right), & i \in N, j \in V \setminus \{S\} \text{ and } j \neq i \\ e_{ij} \frac{p_i^{tr} D_i}{R_{ij}} + e_{ji} \frac{p_i^{re} B_i}{R_{ji}}, & i \in N, j = S \\ 0, & i \in N, j = i \end{cases} \quad (5)$$

3.2 Task cost calculation

When computing task A_i is assigned to a different device for calculation, there must be a corresponding calculation cost. We consider this computational cost as the cost of performing the execution on node j and denote it as Co_i . Various types of collaborative equipment require different calculation costs due to factors such as status and environment, and so the calculation costs of

performing tasks on different nodes are different as well. Therefore, the formula for calculating Co_i is as shown in equation (6).

$$Co_i = \begin{cases} \frac{D_i}{T_i^{\max} - T_{ij}^{tr}} p_j, & i \in N, j \in V, j \neq i \\ 0, & j = i \end{cases} \quad (6)$$

where p_j is the calculation cost of running the unit computing resource when task A_i is executed on node j . It is worth noting that the task does not need to be computationally expensive when executed on a local device.

However, the calculation cost caused by a single task does not represent the pros and cons of the entire unloading decision-making scheme, and the same proportion of the computational cost of the task is obviously unreasonable. Hence, we use $\overline{Co}$ to describe the calculation cost of the entire offloading decision scheme. The calculation formula is as shown in equation (7).

$$\overline{Co} = \sum_{i=1}^n f(D_i) Co_i \quad (7)$$

where $f(D_i)$ is the weight of the calculation cost of each task in the $\overline{Co}$ calculation process. We assume that the larger the amount of data to complete a task, the greater the calculation cost of the task. Based on this kind of thinking, we propose the formula for calculating $f(D_i)$ in equation (8).

$$f(D_i) = \frac{D_i}{\sum_{i \in N} D_i} \quad (8)$$

3.3 Problem formulation

Given that offloading decisions and computational resource allocation are interdependent, and that they both affect quality of service, we jointly optimize them to minimize the sum of calculation costs and energy consumption. The problem is expressed as follows:

$$P : \min_{e, r} (\sum_{i \in N} \sum_{j \in V} (e_{ij} \varepsilon_{ij}^c + e_{ij} \varepsilon_{ij}^{tr}) + n\overline{Co}) \quad (9)$$

subject to

$$C1 : \sum_{j \in V} e_{ij} = 1, \quad \forall i \in N \quad (9-1)$$

$$C2 : \sum_{i \in N} e_{ij} \leq 1, \quad \forall j \in V \setminus \{S\} \quad (9-2)$$

$$C3 : 0 \leq \sum_{i \in N} e_{ij} r_{ij} \leq f_j, \quad \forall j \in V \quad (9-3)$$

$$C4 : 0 \leq \sum_{j \in V} (e_{ij} T_{ij}^{tr} + e_{ij} T_{ij}^c), \quad \forall i \in N \quad (9-4)$$

$$C5 : \sum_{j \in V} (e_{ij} T_{ij}^{tr} + e_{ij} T_{ij}^c) \leq T_i^{\max}, \quad \forall i \in N \quad (9-5)$$

where the variables to be optimized in equation (9) are e and r , which represent the offloading decision and computing resource allocation, respectively. $C1$ guarantees that each task must be executed. $C2$ ensures

that each mobile device can perform at most one task. $C3$ ensures that each device performs its tasks without exceeding its own computing power. $C4$ and $C5$ ensure that each task does not exceed the maximum delay when it is executed. In order to avoid the result of meaningless optimization due to the large difference between the two target orders of magnitude, we plan to select the current maximum and minimum values of the two targets in each iteration. Then we do dynamic min-max normalization for each of these goals.

4 Proposed approach

Aiming at the above optimization problem model, we propose the ANSSA. ANSSA is an adaptive neighborhood search algorithm using SA's Monte Carlo strategy. In the neighborhood search, it has two kinds of destruction operators and three kinds of repair operators to randomly combine to produce a new solution. At the beginning, the probability of selecting each operator is equal, and as the population iteration goes on, the weight of excellent operator selection will increase. In addition, when new solutions are generated, SA's Monte Carlo solution strategy is used to decide whether to accept the new solution or not, so as to avoid falling into local optimality.

4.1 SA

The SA algorithm is a random optimization algorithm based on the Monte Carlo iterative solution strategy. The algorithm uses the Metropolis sampling strategy with probabilistic jump characteristics to perform a random neighborhood search in the solution space, with the temperature decreasing. The sampling process is repeated to finally obtain the global optimal solution or a satisfactory solution to the problem.

Since we solve the problem of minimizing the objective function, according to the Metropolis criterion, the probability p that particles tend to balance at temperature T should be calculated as shown in equation (10).

$$p = \begin{cases} 1, & \Delta f < 0 \\ \exp\left(-\frac{\Delta f}{T}\right), & \Delta f \geq 0 \end{cases} \quad (10)$$

where f is the fitness function of the individual, and Δf can be calculated using equation (11).

$$\Delta f = f(x_{\text{new}}) - f(x_{\text{old}}) \quad (11)$$

The value of T changes with the number of iterations so that the entire system reaches a stable state. The formula for changing the value of T is as shown in equation (12).

$$T = T_{\text{init}} \bullet \alpha^{\text{Gen}} \quad (12)$$

where T_{init} is the initial temperature, α is the temperature decay rate, and Gen is the number of iterations.

4.2 ANS

The neighboring search (NS) algorithm is a common and improved algorithm that finds a better solution by searching the neighborhood of the current solution at each iteration. The neighborhood is implicitly defined using the destroy and repair methods. The destroy method destroys part of the current solution, and the repair method reconstructs the corrupted solution.

On the basis of the NS, the adaptive neighborhood search algorithm (ANS) uses multiple destroy and repair methods in the same search to obtain the neighborhood of the current solution. During the search process, the ANS dynamically adjusts the weights of the various destroy and repair methods to obtain a better neighborhood and solution. The specific process is as follows.

Ω^- and Ω^+ represent a collection of destroy and repair methods, respectively. ρ^- and ρ^+ respectively represent the weight of each destroy and repair method. In the beginning, all methods use the same weight. ϕ_i^- represents the selected probability of destroy method i and ϕ_j^+ represents the selected probability of repair method j . They can be calculated by equation (13).

$$\phi_i^- = \frac{\rho_i^-}{\sum_{k=1}^{|\Omega^-|} \rho_k^-}, \quad \phi_j^+ = \frac{\rho_j^+}{\sum_{k=1}^{|\Omega^+|} \rho_k^+} \quad (13)$$

In general, the greater the weight is, the greater the probability that the method will be selected. The weight size is dynamically adjusted according to the performance of the new solution in the search process of the destroy and repair methods. The update method is calculated as shown in equation (14).

$$\begin{aligned} \rho_i^- &= \lambda \rho_i^- + (1 - \lambda) \psi \\ \rho_j^+ &= \lambda \rho_j^+ + (1 - \lambda) \psi \end{aligned} \quad (14)$$

where $\lambda \in (0, 1)$, and i, j respectively represent the destroy and repair methods selected. ψ represents the evaluation criteria for the new solution, and is evaluated by equation (15).

$$\psi = \max \begin{cases} \omega_1, & \text{if } x_{\text{new}} \text{ is better than } x_{\text{current}} \\ \omega_2, & \text{if } x_{\text{new}} \text{ is worse but it is accepted} \\ \omega_3, & \text{if } x_{\text{new}} \text{ is worse and it is rejected} \end{cases} \quad (15)$$

where $\omega_1 \geq \omega_2 \geq \omega_3 \geq 0$.

4.3 ANSSA

The main frame of the ANSSA is shown in Fig. 2. The main steps of the ANSSA are as follows:

- Step 1 : Initialize the parameters, and obtain a feasible mode matrix for each dimension of the initial individual according to equations $C1 - C5$. Then, we evaluate each mode's fitness value and generate an initial individual for each dimension according to the roulette strategy, which is the initial global optimal solution Global_best. This is shown in Fig. 3.
- Step 2 : According to equation (13), we adopt the roulette strategy and select a random combination of the destruction operator and the repair operator in the strategy pool to perform a neighborhood search on Global_best. This is shown in Fig. 4. Then, we obtain a new solution according to equation (11). In addition, according to equation (14), the update strategy operator selects the weights until a sufficient number of individuals are generated to form the population.
- Step 3 : We allocate resources according to equation (3) and evaluate the fitness value of each individual. Then, we select the individual with the smallest fitness value as the local optimal solution Local_best.
- Step 4 : We compare Global_best with Local_best to select the individuals with smaller fitness values and update Global_best.
- Step 5 : Return to Step 2 and update the value of T according to equation (12) until the number of loops reaches the number of iterations.
- Step 6 : Output the current optimal solution Global_best and its fitness value.

The core mechanism of the ANSSA is illustrated in Fig. 3 and Fig. 4. Fig. 3 shows the initial unloading strategy development process for five tasks. After obtaining the feasible candidate pattern sets for each of the five tasks according to $C1 - C5$, we determine the task priority according to the size of the feasible candidate pattern set, and the execution mode is selected

Fig. 2: The framework of the ANSSA

Fig. 3: Illustration of the generation strategy of the initial solution

Fig. 4: Illustration of a neighborhood search process

first for tasks with a small set. It is worth noting that each mobile device can only be used once. In Fig. 3, the feasible set of U4 is the smallest, and so this execution mode is preferred. There are two ways to select the execution mode, namely, the roulette strategy or the sum of both the total energy consumption and calculation cost minimum mode. The execution mode of the above five tasks is selected according to the above two selection modes. According to this rule, the initial solution is generated until all task assignment patterns are calculated.

Fig. 4 describes the neighborhood search process using an example. First, the destruction operator that destroys two tasks according to their priority is selected from the destruction strategy set. U4 and U5 have higher priorities and are chosen to destroy the values corresponding to U4 and U5. Then, select the repair operator that conducts random repairing from the repair strategy set and assign U4 and U5 in an unordered manner. In this example, U5 is first extended to the base station, and then U4 is extended to mobile device 3. It is worth noting that there are many destruction operators in destruction strategy set. The repair operator in the repair strategy set is based on the number of tasks destroyed, which are randomly repaired, repaired by priority or regenerated, as shown in Fig. 3.

5 Experimental studies

The purpose of the experiments in this section is to verify the effectiveness of the ANSSA. The data set in (Huang et al., 2019) is used in the experiment, and includes the channel gain, computing power, task data size and other related information of one MEC server and two instance suites. Based on the data set, the calculation cost for the collaborative device to complete the task running unit computing resource is randomly generated.

In this paper, two indicators are used to measure the quality of the experimental results, which are the

Table 1: Experimental Environment

Software and Hardware	Specific Information
Running Software	MATLAB R2018b
Processor Frequency	2.20 GHz
RAM	8 GB
CPU	i7-8750

run time (RT) and total consumption (TC). Note that all algorithms are implemented in MATLAB R2018b using a personal computer with an i7-8750 CPU running at 2.20 GHz (processor) and 8 GB RAM. The details are shown in table I. The run time is the time it takes for the algorithm to iterate 300 times and finally get the optimal solution. The Total consumption is the sum of the energy consumption and the calculation cost. The experimental results are the average of the results obtained after 20 independent operations.

As shown in Fig. 5 and Fig. 6, in order to verify the validity of the ANSSA algorithm, the ANSSA is compared with the BIJOR and IANSSA using two example suites. The Bilevel Optimization Approach for Joint Offloading Decision and Resource Allocation (BIJOR) is a two-layer optimization method that was combined with ant colony algorithm in (Huang et al., 2019), and the Init_ANSSA (IANSSA) is basically the same as the ANSSA. The difference is that we add BIJOR's individual generation algorithm as a repair operator to the selection pool of the ANSSA repair operators, thereby enriching the selection pool of the ANSSA repair operators, and diversifying the choice of the ANSSA to solve problems.

In Fig. 5 and Fig. 6, the IANSSA has the lowest average RT for the two example suites. In other words, the solution speed is the fastest, but the average TC of the IANSSA is higher than those of the BIJOR and ANSSA. In addition, the average TC of the ANSSA is the smallest in all cases. In other words, the solution is the best, and the average RT is not very high, which means that it is an excellent algorithm. When the number of users is 200, the average TC of the IANSSA is 1.07 times that of the BIJOR and 1.10 times that of the ANSSA, but the average run time is 0.22 times that of the BIJOR and 0.38 times that of the ANSSA. When the number of users is 300, the average total cost using the IANSSA algorithm is 1.01 times that of the BIJOR and 1.04 times that of the ANSSA, but the average run time is 0.22 times that of the BIJOR and 0.33 times that of the ANSSA. However, as the number of users increased to 400, the performance of the IANSSA solution declined rapidly, although its time is still the lowest. Obviously, the IANSSA has limitations. The performance of the BIJOR is slightly worse than that of the

Table 2: Comparison of the average TCs and SRs of the ANSSA-LP and ANSSA for the higher number of users instance suites

n	ANSSA-LP		ANSSA	
	SR	Average TC	SR	Average TC
50	100.0%	33.04856	100.0%	31.43553
80	100.0%	47.46972	100.0%	45.59416
100	20.0%	\	100.0%	103.15806
120	20.0%	\	100.0%	145.36302
150	5.0%	\	100.0%	181.77861
200	10.0%	\	100.0%	324.2274
300	0.0%	\	100.0%	477.09497
400	0.0%	\	100.0%	772.85684

ANSSA, but in all cases, the average RT is the largest, and it is impossible for the method to quickly obtain a better solution, which may lead to the adverse consequence of the user having left the effective range when getting a good solution. The ANSSA combines the evolutionary strategies of the two algorithms and adopts an adaptive mechanism to improve the performance of the solution while maintaining a short running time and better adaptability to multiple scenarios.

In Fig. 7, the initial solution quality of the ANSSA is not high, but because of the diversity of the evolutionary operators and the adaptive mechanism, the ANSSA can quickly converge to the global optimal solution. However, the IANSSA can only converge to the local optimal solution. The quality of the initial solution of the BIJOR is high, but its population diversity has limitations, and it is difficult to converge to the global optimal solution. With the expansion of the number of users, this shortcoming is even more obvious. The above phenomenon can be explained as follows: the ANSSA uses a variety of neighborhood search operations to obtain a richer population based on the initial solution, but the quality of the initial solution has a greater impact on the ANSSA; therefore, the BIJOR is used as an evolutionary operator of the ANSSA. The ANSSA can not only obtain a higher quality initial solution but also fully search the solution space, escape the local optimal solution, and make the selection of multiple evolutionary operators more rational in the continuous search through its own adaptive mechanism. Therefore, it obtains the best solution performance.

In this paper, we have prioritized each task for the construction of the unloading decision. To investigate the performance impact of the sorting strategy on the offloading decision build, we derived the ANSSA-LP variant from the ANSSA using a random sorting strategy when assigning patterns to each task. As shown in table II, the SRs are both 100% when $n=50$ and 80. However, the ANSSA with the priority sorting strategy

Fig. 5: Average TCs and RTs for the BIJOR, IANSSA, and ANSSA on a low user count instance suite

Fig. 6: Average TCs and RTs of the BIJOR, IANSSA, and ANSSA on high user count instance suites

Fig. 7: Evolution of the average TCs of the BIJOR, IANSSA, and ANSSA algorithms for three different numbers of users

is always superior to the ANSSA-LP with respect to the Average TC. When the number of tasks is greater than 80, there is a significant performance difference between the ANSSA and ANSSA-LP. It can be seen that priority sorting can better generate feasible solutions, which is conducive to finding the optimal solution of the problem.

6 Conclusion

Aiming at the bottleneck problem of an MEC server in the field of edge mobile computing, we combine collaborative computing with mobile edge computing to propose a multiuser collaborative mobile edge computing model. Based on collaborative computing, a new indicator – the calculation cost – is first proposed to compensate for using other mobile devices, thereby further improving the rationality of the model. To solve this model, we propose the ANSSA, which adopts an adap-

tive mechanism to update the probability of selection according to historical experience and environmental perception. Finally, the ANSSA is compared with two other algorithms using two example suites. The experimental results show that the ANSSA can obtain satisfactory results in complex scenarios.

Acknowledgements We would like to thank the referees for their valuable comments and suggestions.

Author Contributions All authors have contributed to this research equally.

Funding This paper was supported in part by Project funded by China Postdoctoral Science Foundation under Grant 2020M671552, in part by Jiangsu Planned Projects for Postdoctoral Research Funds under Grant 2019K223, in part by NUPTSF (NY220060), in part by the Opening Project of Jiangsu Key Laboratory of Data Science and Smart Software (No.2020DS301), in part by Natural Science Foundation of Jiangsu Province of China under Grant BK20191381, in part by the National Natural Science Foundation of China under Grant 61772286, Grant 61771258 and Grant 61876089.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human or animals performed by any of the authors.

References

- Abbas N, Zhang Y, Taherkordi A, Skeie T (2017) Mobile edge computing: A survey. *IEEE Internet of Things Journal* 5(1):450–465
- Cao X, Wang F, Xu J, Zhang R, Cui S (2018) Joint computation and communication cooperation for energy-efficient mobile edge computing. *IEEE Internet of Things Journal* 6(3):4188–4200
- 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
- Gumaei A, Al-Rakhami M, AlSalman H, Rahman SMM, Alamri A (2020) DI-har: Deep learning-based human activity recognition framework for edge computing. *CMC-COMPUTERS MATERIALS & CONTINUA* 65(2):1033–1057
- Guo J, Song Z, Cui Y, Liu Z, Ji Y (2017) Energy-efficient resource allocation for multi-user mobile edge computing. In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*, IEEE, pp 1–7
- Huang PQ, Wang Y, Wang K, Liu ZZ (2019) A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. *IEEE Transactions on Cybernetics* doi: <https://dx.doi.org/10.1109/TCYB.2019.2916728>
- Liu M, Liu Y (2017) Price-based distributed offloading for mobile-edge computing with computation capacity constraints. *IEEE Wireless Communications Letters* 7(3):420–423
- Liu P, Ren H, Shi X, Li Y, Cai Z, Liu F, Zeng H (2020) Motransframe: Model transfer framework for cnns on low-resource edge computing node. *CMC-COMPUTERS MATERIALS & CONTINUA* 65(3):2321–2334
- Mao Y, Zhang J, Letaief KB (2016) Dynamic computation offloading for mobile-edge computing with energy harvesting devices. *IEEE Journal on Selected Areas in Communications* 34(12):3590–3605
- Pu L, Chen X, Xu J, Fu X (2016) D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted d2d collaboration. *IEEE Journal on Selected Areas in Communications* 34(12):3887–3901
- Qasim A, Kazmi SAR (2019) Formal modelling of real-time self-adaptive multi-agent systems. *Intelligent automation and soft computing* 25(1):49–63
- Qian C, Li X, Sun N, Tian Y (2020) Data security defense and algorithm for edge computing based on mean field game. *Journal of Cybersecurity* 2(2):97
- Ti NT, Le LB (2017) Computation offloading leveraging computing resources from edge cloud and mobile peers. In: *2017 IEEE International Conference on Communications (ICC)*, IEEE, pp 1–6
- Tran TX, Pompili D (2018) Joint task offloading and resource allocation for multi-server mobile-edge computing networks. *IEEE Transactions on Vehicular Technology* 68(1):856–868
- WANG J, YU Sj (2011) Blind detection based on simulated annealing chaotic particle swarm optimization. *Computer Technology and Development* 21(1):35–37
- Wei F, Chen S, Zou W (2018) A greedy algorithm for task offloading in mobile edge computing system. *China Communications* 15(11):149–157
- Wei X, Liu J, Wang Y, Tang C, Hu Y (2021) Wireless edge caching based on content similarity in dynamic environments. *Journal of Systems Architecture* 115:102000, doi: <https://doi.org/10.1016/j.sysarc.2021.102000>
- Xue Y, Tang T, Pang W, Liu AX (2020) Self-adaptive parameter and strategy based particle swarm optimization for large-scale feature selection problems with multiple classifiers. *Applied Soft Computing* 88:106031
- Zhang K, Mao Y, Leng S, Zhao Q, Li L, Peng X, Pan L, Maharjan S, Zhang Y (2016) Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks. *IEEE access* 4:5896–5907
- Zheng H, Hu Xb, Zheng Mm, Liu Rj (2013) An improved hybrid algorithm based on particle swarm optimization and simulated annealing and its application. *Comput Technol Dev* 23(7):26–30