Feng *et al. EURASIP Journal on Advances in Signal Processing* (2021) 2021:36
https://doi.org/10.1186/s13634-021-00751-5

EURASIP Journal on Advances in Signal Processing

RESEARCH Open Access

Optimizing computation offloading strategy in mobile edge computing based on swarm intelligence algorithms

Siling Feng[1*], Yinjie Chen[1], Qianhao Zhai[2], Mengxing Huang[1,3*] and Feng Shu[1]

*Correspondence:
fengsiling2008@163.com;
huangmx09@hainanu.edu.cn
[1]School of Information and Communication Engineering, Hainan University, No. 58 Renmin Avenue, 570228 Haikou, China
[3]State Key Laboratory of Marine Resource Utilization in the South China Sea, Hainan University, No. 58 Renmin Avenue, 570228 Haikou, China
Full list of author information is available at the end of the article

Abstract

As the technology of the Internet of Things (IoT) and mobile edge computing (MEC) develops, more and more tasks are offloaded to the edge servers to be computed. The offloading strategy performs an essential role in the progress of computation offloading. In a general scenario, the offloading strategy should consider enough factors, and the strategy should be made as quickly as possible. While most of the existing model only considers one or two factors, we investigated a model considering three targets and improved it by normalizing each target in the model to eliminate the influence of dimensions. Then, grey wolf optimizer (GWO) is introduced to solve the improved model. To obtain better performance, we proposed an algorithm hybrid whale optimization algorithm (WOA) with GWO named GWO-WOA. And the improved algorithm is tested on our model. Finally, the results obtained by GWO-WOA, GWO, WOA, particle swarm optimization (PSO), and genetic algorithm (GA) are discussed. The results have shown the advantages of GWO-WOA.

Keywords: Mobile edge computing, Computation offloading, Grey wolf optimizer, Whale optimization algorithm

1 Introduction

With the development of IoT technology, mobile devices (MDs) are commonly used to collect data and process them. These devices are usually made to be small with limited computing resources and energy supply. However, in some applications, the computation tasks are so complicated that the processing unit on mobile devices may need a long time to deal with, increasing concern about its high energy consumption. Mobile cloud computing (MCC) is proposed to break through the barrier between the request for complex applications and restricted resources. In general, MCC application scenarios, computation tasks are performed on the central cloud, which has enormous storage space and rich computational resources [1]. Although MDs obtained the ability to process complex computation tasks with locally low energy consumption through this method, MCC lacks latency [2]. The centralized servers are usually remote from MDs. With the development

© The Author(s). 2021 **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

of the Internet and fifth-generation mobile networks (5G), more applications perform real-time processes and require low latency for MCC.

The generation of MEC solves the problems. In MEC scenarios, edge servers are distributed nearly everywhere (commonly with the wireless base station in 5G networks). As the servers are physically closer to MDs, it can effectively reduce latency and reduce the energy consumption caused by data transmission. Compared with a centralized server cluster, MEC servers do not have that rich resources. It is not a problem because MEC servers only provide service to a specific area. The capacity can be flexibly adjusted to suit the actual load. Besides, MDs may not always generate heavy computation tasks, and it will cause a waste of resources if the tasks are always sent to be processed on servers, no matter how much it is. With the feature of MEC, MDs are more flexible in dealing with the tasks, with more optional choices on whether to offload the computation tasks or not and how much to offload the computation tasks. The offloading decisions have significant impacts on the quality of service (QoS). With the aforementioned backgrounds, computation offloading decision is an essential branch of MEC, receiving more and more attention.

The target of computation offloading can be comprehensive, including time, energy, cost, etc. The time target aims to reduce the latency, while the energy target aims to reduce the power consumption. The cost target can be considered as two sides. One is the cost of transmitting the data, and the other is the cost of edge computation resources. Some methods only consider a single target. For example, [3, 4] only consider the time target, and [5, 6] only consider the energy target. Some methods consider two targets, like [7, 8]. The research that considers three or more targets does exist, but it is seldom compared with those considering one or two targets. As a result, research on more than two targets is desperately needed.

Once the model of computation offloading is proposed, the next step is to find out the proper and effective method to obtain the computation offloading decisions. The decisions obtained should get the best result of the model with given conditions. Swarm intelligence is the collective intelligence behavior of self-organized and decentralized systems [9]. Moreover, swarm intelligence algorithms are a kind of algorithm that has attracted interest from many researchers in various kinds of fields. The many researches and applications of swarm intelligence optimization algorithms show that they can solve the defined computation offloading model.

As the multi-target optimization problem is an NP-hard problem [10], an accurate method is not rather than suitable. This kind of optimization problem is more suitable to use a non-accurate method [11–13], such as evolution algorithms, swarm intelligence algorithms, et cetera. In this paper, swarm intelligence algorithms are considered to be used to solve the problem.

Deng et al. [14] present a computation offloading model with 0–1 planning and weight improvement and solve the model with PSO algorithm. Pham et al. [15] study the computation offloading in non-orthogonal multiple access (NOMA)-based multi-access edge computing systems. Moreover, WOA is used to optimize the joint optimization problem of offloading decision, subchannel assignment, transmit power, and computing resource allocation. Pham et al. [16] present the adoption of WOA to solve various resource allocation problems.

In works [14–16], none of them covers the problem with both multiple targets on computation offloading problems and state-of-the-art optimization algorithms. The main contribution of this paper can be summarized as follows. First, the computation offloading model in this paper considers three targets, time delay, energy consumption, and service price, and the model is improved by using normalization. Then, a swarm intelligence algorithm named GWO is applied to solve the proposed model. Next, according to the existing performance of WOA, an improvement that combines WOA with GWO is proposed, and it is given the name GWO-WOA. The whale optimization algorithm has only one leader solution when searching for the best solution in the set. This characteristic of WOA can cause it easily converge at the local optima. While GWO has three leaders during the searching process, it has less possibility to fall into local optima. We can use it to improve the searching progress of WOA and improve the performance of the original WOA. Finally, GWO-WOA is applied to solve the proposed model. The results show the excellent performance of the proposed model and GWO-WOA.

The rest of this paper is organized as follows. In Section 2, some related works are represented and discussed. In Section 3, the system model are represented, including the local computing model, edge computing model, service price model, and problem formulation. In Section 4, two solutions to solve the model are represented in detail. In Section 5, the results of experiments are shown in the form of tables and figures and are discussed. In Section 6, the study has been concluded.

2 Related works

Both computation offloading strategies and swarm intelligence algorithms are attractive research directions. Some excellent researches have been done in recent years.

To offload the computing task, computation offloading can be divided into binary computation offloading and partial computation offloading. The former means MDs can only fully offload computational tasks to the edge servers or compute them locally. The latter is more flexible than the former, which means tasks can be dealt with partly, not wholly. Zhu et al. [17] have employed the game theory to optimize the multi-user binary computation offloading problem. A Q-learning based method is applied to solve the binary computation offloading problem on the work of Jiang et al. [18]. Zhao et al. [19] proposed a partial computation offloading strategy using reinforcement learning to reach the minimum cost of the system.

For the number of objectives the computation offloading strategy involved, computation offloading can be divided into single objective computation offloading and many objectives computation offloading. Miao et al. [20] propose a computation offloading and task migration algorithm to reduce the processing time of applications. Jiang et al. [21] studied computation-intensive and delay-sensitive task scheduling, where an efficient task scheduling algorithm is developed to solve the optimization problem. Huang et al. [22] use a multi-objective method to solve the problem considering time consumption and energy consumption. Yan et al. [23] have worked on the joint task offloading and resource allocation problem by considering both the energy consumption and execution time.

As for the perspective of device amount considered in the model, computation offloading can be separated as single-user and multi-users. Labidi et al. [24] discussed the balance between shortening the execution time and extending the mobile device's battery

life under single-user scenarios. And You et al. [25] solves the problem of computation offloading in a multi-user scenario.

From the three perspectives, it can be known that the work in this paper, which uses a swarm intelligence algorithm to solve multi-device three targets is necessary.

3 Methods

3.1 System model

This paper discusses the scenario that MDs in a particular area offload their computational tasks to specific edge servers. Each MDs has a computational-intensive task need to be computed. The set of MDs can be denoted as $N = \{1, 2, \ldots, n\}$. $C = \{c_1, c_2, \ldots, c_n\}$ is used to represent the CPU cycles needed to finish the task, and $D = \{d_1, d_2, \ldots, d_n\}$ is used to represent the data size of computational tasks, where i is corresponding to the mobile device i in MDs set N. The combination of set C and set D are to describe the tasks on each MD. Communication, including computational data transmission between mobile devices and edge servers, is performed through the wireless access point. It is assumed that each task can be partial or fully offload to the edge server. The set $X = \{x_1, x_2, \ldots, x_n\}$ is used to represent the offloading decisions, where x_i belongs to $[0, 1]$. If $x_i = 0$, it means mobile devices i compute the task through its CPU locally. On the contrary, the mobile device i completely offload its task to be computed at the edge servers if $x_i = 1$. If $x_i > 0$ and $x_i < 1$, it means MD i offloads $x_i \times 100\%$ of tasks to be computed at the edge servers, and the rest $(1 - x_i) \times 100\%$ is computed at local. In the scenario, the computation resource capacity on edge servers is considered. The computation offloading model is a joint optimization of time delay, energy consumption, and price for edge service.

3.1.1 Local computing model

For the situation that computes the task at local, the model can be described in this section. We let t_i^l as the local execution time, which only includes the processing time for the local CPU and e_i^l as the corresponding energy consumption of processing the task. F_i^l is denoted as the maximal CPU cycle frequency of device i due to its hardware. f_i^l is denoted as the current CPU cycles available for the task according to the run-time situation of mobile device i. When mobile device i process its task locally, the time delay t_i^l can be defined as:

$$t_i^l = \frac{c_i}{f_i^l} \tag{1}$$

And the energy consumption can be expressed as:

$$e_i^l = \kappa \left(f_i^l\right)^2 c_i \tag{2}$$

where κ means the effective switched capacitance that is determined by the chip architecture, according to the reference [26], κ is set as 10^{-27} in this paper.

3.1.2 Edge computing model

The model can be described in this section for the situation that computes the task at the edge server.

As the communication is through the wireless channel, the communication rate should be considered. W is defined as the bandwidth of the wireless channel, assuming that it

would be equally allocated to mobile devices if more than one device chooses to offload the task simultaneously. Under this setting, θ_i is the wireless channel bandwidth allocated to mobile device i. The communication rate of mobile device i can be denoted as [14]:

$$R_i = r_i\theta_i = W \log\left(1 + \frac{p_i h_i}{W N_0}\right)\theta_i \tag{3}$$

where p_i represents the transmission power of mobile device i, h_i represents the channel gain of mobile device i, N_0 denotes the background channel noise.

Under this circumstance, the time delay can be divided into two parts: transmission time and process time. t_i^o is used to represent the transmission time, and it can be defined as:

$$t_i^o = \frac{d_i}{R_i} \tag{4}$$

The whole computing resources of the edge servers can be represented as F. And f_i^e denotes the CPU cycle frequency allocated to the mobile device i to finish its task at the edge server. t_i^e denotes the processing time needed on the edge server for the task from mobile device i. It can be defined as:

$$t_i^e = \frac{c_i}{f_i^e} \tag{5}$$

The time for computation results to be transmitted back to the mobile device is ignored due to the data size of the result is much smaller. The total time for the mobile device i to complete its task fully through the edge server should be calculated by:

$$t_i^p = t_i^o + t_i^e \tag{6}$$

And corresponding energy consumption e_i^p can be defined as:

$$e_i^p = P_i^o t_i^o + P_i^e t_i^e \tag{7}$$

where P_i^o is the power needed to transmit data from mobile device i through the wireless access point, and P_i^e is the power when the mobile device i is waiting for the result.

3.1.3 Edge service pricing model

The price of servicing charging mainly has two kinds of patterns. One is charging for the usage of time, and the other is charging for the resource usage. In this model, charging price based on resource usage is considered. In real life, the price is set on a certain unit. So, the baseline CPU frequency cycles f_{base} is defined. And according to the baseline set, the charge for the baseline V_{base} is defined as 1. The cost incurred for the task of each mobile device can be defined as:

$$u_i = t_i^e \times V_{base} \times \frac{f_i^e}{f_{base}} \tag{8}$$

3.1.4 Problem formulation

In the problem, it is assumed that n mobile devices are included. Moreover, each device has a different amount of tasks, which means that each device's workload varies. The decision set X is calculated according to the given set of computation complexity C and data size D. The model should consider time delay, energy consumption, and pricing. However, these three targets describe different metrics, and they can not be simply added to form the final target function. Otherwise, it may cause problems. For example, the

quantity difference may force the target function only to focus on a specific target. To solve the problem mentioned, normalization is applied in the model.

The total time latency can be calculated as:

$$T = \sum_{i=1}^{n} \frac{\left[(1-x_i)\, t_i^l + x_i t_i^p\right] - T_{\min}}{T_{\max} - T_{\min}} \tag{9}$$

where $T_{\min}$ means the minimum time delay calculated in the mobile device set, and $T_{\max}$ means the maximum time delay calculated in the mobile device set.

The total energy consumption can be calculated as:

$$E = \sum_{i=1}^{n} \frac{\left[(1-x_i)\, e_i^l + x_i e_i^p\right] - E_{\min}}{E_{\max} - E_{\min}} \tag{10}$$

where $E_{\min}$ means the minimum energy consumption calculated in the mobile device set, and $E_{\max}$ means the maximum energy consumption calculated in the mobile device set.

The total price of edge service can be calculated as:

$$U = \sum_{i=1}^{n} \frac{x_i u_i - U_{\min}}{U_{\max} - U_{\min}} \tag{11}$$

The improved calculation method has eliminated the influence of dimensions and makes the objective function easier to reflect the change of the result when adjusting the decision. As a consequence, the objective function can be expressed as below:

$$\begin{aligned} Q &= T + \eta E + \gamma U \\ &= \sum_{i=1}^{n} \frac{\left[(1-x_i)\, t_i^l + x_i t_i^p\right] - T_{\min}}{T_{\max} - T_{\min}} + \eta \sum_{i=1}^{n} \frac{\left[(1-x_i)\, e_i^l + x_i e_i^p\right] - E_{\min}}{E_{\max} - E_{\min}} \\ &\quad + \gamma \sum_{i=1}^{n} \frac{x_i u_i - U_{\min}}{U_{\max} - U_{\min}} \end{aligned} \tag{12}$$

where η and γ is used as the coefficients. The coefficients are used to adjust the relationship of the three targets, which can be seen as the weights of targets in the final formulation. In the equation, the time latency target is regarded as a baseline whose coefficient is 1. The coefficients of the other two targets are adjusted, and the proportion of these three parts becomes different. In this way, the wanted weight consideration for the three targets is achieved.

The optimization problem to be solved can be given by:

$$\begin{aligned} &\min_{x_i, \theta_i, f_i^e} Q \\ s.t.\ &0 \le f_i^e \le x_i F,\ \forall i \in N \\ &\sum_{i=1}^{n} f_i^e \le F,\ \forall i \in N \\ &0 \le \theta_i \le x_i L,\ \forall i \in N \\ &\sum_{i=1}^{n} \theta_i \le L,\ \forall i \in N \\ &x_i \in [0, 1],\ \forall i \in N \end{aligned} \tag{13}$$

The optimization problem has three targets, including time latency, energy consumption, and service price. The goal of the optimal problem is to reach the minimum value of the Q function under limited conditions.

4 Problem solutions

4.1 Grey wolf optimizer

Grey wolf optimizer (GWO) [27] is a swarm intelligence algorithm inspired by the hunting pattern and the social hierarchy of grey wolves. Grey wolves mostly live in a pack with a strict social dominant hierarchy. The pack of wolves can be categorized into four groups: alpha, beta, delta, and omega. Alpha wolves are the leader of the pack of wolves; beta wolves are subordinate wolves helping alphas; delta wolves are responsible for watching the boundaries of territory and warning for dangers; and omega wolves play the role of scapegoat, which dominated by the other three groups of wolves. In the algorithm of GWO, the fittest solution is considered as the alpha, while the second-best and the third-best solutions are considered as beta and delta, respectively. The rest of the solutions are regarded as omega. And the hunting behavior of grey wolves is abstracted to three stages in the algorithm: encircling prey, hunting, attacking prey, and search for prey.

For the encircling stage, the positions of the wolves can be updated by [27]:

$$\overrightarrow{D} = \left| \overrightarrow{C} \cdot \overrightarrow{X_p}(t) - \overrightarrow{X}(t) \right| \tag{14}$$

$$\overrightarrow{X}(t+1) = \overrightarrow{X_p}(t) - \overrightarrow{A} \cdot \overrightarrow{D} \tag{15}$$

where t is the current iteration, $\overrightarrow{X}$ is the position vector of a grey wolf, $\overrightarrow{X_p}$ is the position of the prey. $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors which can be calculated as:

$$\overrightarrow{A} = 2 \cdot \overrightarrow{a} \cdot \overrightarrow{r_1} - \overrightarrow{a} \tag{16}$$

$$\overrightarrow{C} = 2 \cdot \overrightarrow{r_2} \tag{17}$$

where the components of $\overrightarrow{a}$ are linearly decreased from 2 to 0 through the iterations and $\overrightarrow{r_1}$, $\overrightarrow{r_2}$ are two random vectors whose value is in $[0, 1]$.

For the hunting stage, the positions of the wolves can be updated by [27]:

$$\overrightarrow{D_\alpha} = \left| \overrightarrow{C_1} \cdot \overrightarrow{X_\alpha} - \overrightarrow{X} \right|, \overrightarrow{D_\beta} = \left| \overrightarrow{C_2} \cdot \overrightarrow{X_\beta} - \overrightarrow{X} \right|, \overrightarrow{D_\delta} = \left| \overrightarrow{C_3} \cdot \overrightarrow{X_\delta} - \overrightarrow{X} \right| \tag{18}$$

$$\overrightarrow{X_1} = \overrightarrow{X_\alpha} - \overrightarrow{A_1} \cdot \overrightarrow{D_\alpha}, \overrightarrow{X_2} = \overrightarrow{X_\beta} - \overrightarrow{A_2} \cdot \overrightarrow{D_\beta}, \overrightarrow{X_3} = \overrightarrow{X_\delta} - \overrightarrow{A_3} \cdot \overrightarrow{D_\delta} \tag{19}$$

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}}{3} \tag{20}$$

With the existing applications of GWO, it has been proved to have superior exploitation, good exploration ability, and high local optima avoidance. GWO shows the potential to solve the optimization model proposed. The core pseudocode of the GWO algorithm can be shown in Algorithm 1.

4.2 Improved WOA with GWO

Similar to GWO, WOA is another kind of swarm intelligence algorithm [28], imitating the hunting behavior of humpback whales. The process of a whale optimization algorithm can be separated into three stages: encircling prey, bubble-net attacking, and searching for

Algorithm 1 Pseudocode of GWO algorithm

Initialize the grey wolf population X_i ($i = 1, 2, \ldots, n$)
Initialize $\vec{a}$, $\vec{A}$, and $\vec{C}$
Calculate the fitness value of each search agent
Select X_α, X_β, X_δ from solutions according to the fitness values
while $t <$ Max iterations number **do**
  for each search agent **do**
    Update the position of the current search agent by using (20)
  end for
  Update $\vec{a}$, $\vec{A}$, and $\vec{C}$
  Calculate the fitness values of each search agent
  Update X_α, X_β, X_δ
  $t = t + 1$
end while
Return X_α

prey. There have many successful applications of WOA, solving problems in many fields. It shows well balance between exploration and exploitation and has efficient performance against standard algorithms. However, despite the good points of WOA, the algorithm also appears to have some advantages in the application scenarios, like the low efficiency in convergence caused by using a single parameter [29], the failure to jump out from local optima [30]. So, an improvement for WOA is desperately needed.

Considering and comparing the thought and method of WOA with those of GWO, the social hierarchy in GWO is introduced to WOA in this paper, with the purpose of improving the ability to search for global optima and improving the avoidance of falling into local optima. Correspondingly, the process for updating the position of the search agents is modified to suit the introduction of hierarchy. The improved WOA is called GWO-WOA.

The encircling stage of WOA is the same as GWO, while the bubble-net attacking method of WOA is different from the hunting stage of GWO. The WOA has a random mechanism with some random variables. $\vec{A}$ is a random vector that can be calculated by Equation 16. p is a random number in $[0, 1]$, and l is a random number in $[-1, 1]$. When $p < 0.5$ and $\left|\vec{A}\right| \geq 1$, the position of the search agents can be updated by using Equation 15. When $p < 0.5$ and $\left|\vec{A}\right| < 1$, the updating method of the hunting stage of the GWO algorithm is used, replacing the original method that only updates the single leader search agent. When $p \geq 0.5$, the position of the search agent can be updated by [28]:

$$\vec{D'} = \left|\vec{X_p}(t) - \vec{X}(t)\right| \tag{21}$$

$$\vec{X}(t+1) = \vec{D'} \cdot e^{bl} \cdot \cos(2\pi l) + \vec{X_p}(t) \tag{22}$$

The core pseudocode of GWO-WOA can be shown in Algorithm 2.

Algorithm 2 Pseudocode of GWO-WOA algorithm

Initialize the whale population X_i ($i = 1, 2, \ldots, n$)
Calculate the fitness of each search agent
Initialize $\vec{a}$, $\vec{A}$, $\vec{C}$, l, and p
Select X_α, X_β, X_δ from solutions according to the fitness values
while ($t <$ maximum number of iterations) **do**
  for each search agent **do**
    Update $\vec{a}$, $\vec{A}$, $\vec{C}$, l, and p
    if ($p < 0.5$) **then**
      if ($\left|\vec{A}\right| \geq 1$) **then**
        Update the position of the current search agent by using (15)
      else
        Update the position of X_α, X_β, X_γ by using (18)(19)(20)
      end if
    else
      Update the position of the current search agent by using (22)
    end if
  end for
  Check if any search agent goes beyond the search space and amend it
  Calculate the fitness of each search agent
  Update X_α, X_β, X_γ if there is a better solution
  t=t+1
end while

5 Results and discussion

This section will carry out numerical experiments based on the system model above and the algorithm proposed. The algorithms are coded in MATLAB 2021a, and all tests are performed on a PC with a Windows 10 operating system and 8 GB of RAM.

There are edge servers in the center of the service area and some mobile devices whose amount can be adjusted in the simulation scenario. Each mobile device is distributed randomly in the service area. As for each mobile device, it has its own task need to be computed, and the data size and the needed CPU cycles of the task are randomly generated, specifically $d_i \sim N(1000, 100)$ and $c_i \sim N(400, 100)$. The computation resource of the edge servers is F = 40 GHz, and the CPU frequency of the mobile device is randomly from 0.5 to 1 GHz. The power when transmitting data P_i^o is set as 100 mW, and the power when waiting for the result P_i^e is set as 10 mW. The baseline resource of the edge server for charging is set as 1 GHz, and the price for the baseline is set as 1.

Under this setting, we need to perform some experiments to evaluate our algorithm. The goal for our improvement on the algorithm is to propose an algorithm with better performance in the application. As the multi-target problem is formulated into a single target one, we should consider the performance from the convergence and stability. Besides, some standard methods also should be included as comparisons.

The GWO applied is used as a method in the results. And the GWO-WOA is also used to obtain the result. Moreover, WOA is also used to get the results to compare whether the improved method is valid. PSO [31], one common swarm intelligence algorithm, and one

traditional evolution algorithm, GA [32], are included as comparisons for comprehensive analysis.

We choose some indicators to evaluate the performance of these methods. The first one is the value of the Q function, which is the final target, no doubt should be included. The lower the value of the Q function is, the better the method is. The second one is the processing time. In the scenario of computation offloading, offload decisions should be made as quickly as possible, or it will lose significance. The third one is the stability of results with the same inputs. Intelligence algorithms have uncertainty due to the principle of these algorithms, and the results can be affected by this kind of uncertainty. However, the influence of uncertainty on the results should be reduced as much as possible. It can be seen that the algorithm is not stable if the result from the same input has a significant difference each time. For the last, the convergence curve also should be investigated.

The values of the Q function obtained under different device amount settings are not comparable because more mobile devices mean introducing more data, which some values may large and cause normalization to become smaller.

From Table 1, we can know that our method GWO-WOA is the suboptimal value of the function and also is suboptimal in processing time when the device amount is 60. Although GA has the best value of the function, it is placed last in process time, which is almost ten times longer than other methods, while the suboptimal value of the function is just slightly worse than it. As it is known that offloading decisions should be made as soon as possible, the disadvantage of GA in process time has covered the slight advantage of GA in the optimized value of the function. It also can be obtained from Table 1 that GWO has the third-lowest value of the function with the fourth-lowest process time, and WOA has the fourth-lowest value of the function with the third-lowest in process time. It can be summarized that GWO is lean on the speed of processing, and WOA is lean on the result of processing. It is reasonable that GWO-WOA has included both advantages of GWO and WOA. And the result of GWO-WOA has shown its success.

By analyzing the result when the device amount is 90 in Table 1, it can be known that GWO-WOA takes the best optimized value of the function with the suboptimal

Table 1 The results of the algorithms on the proposed model

Device amount	Algorithm	Value of function			Processing time (unit:s)		
		Min	Mean	Max	Min	Mean	Max
60	GWO-WOA	13.0440	14.5241	16.6638	0.1236	0.1336	0.1499
	GWO	14.1564	14.9303	15.8786	0.1320	0.1390	0.1739
	WOA	13.6214	15.5889	17.9665	0.0934	0.1021	0.1150
	PSO	15.5270	17.1228	18.7959	0.2721	0.2887	0.3250
	GA	13.5747	14.2749	15.4485	2.2403	2.3562	2.5156
90	GWO-WOA	8.3972	9.5985	10.6434	0.1722	0.1839	0.2046
	GWO	9.0704	9.7819	10.4810	0.1898	0.2001	0.2238
	WOA	9.1978	10.8786	11.9473	0.1278	0.1376	0.1571
	PSO	10.4215	11.7541	12.7919	0.2829	0.3032	0.3460
	GA	10.3010	11.3086	12.6800	2.2827	2.4174	2.6159
120	GWO-WOA	8.9841	9.7230	10.5497	0.2336	0.2477	0.2779
	GWO	9.0264	9.8471	10.4858	0.2473	0.2618	0.2957
	WOA	9.5342	10.5831	11.8731	0.1697	0.1855	0.2351
	PSO	10.4320	11.5799	12.6505	0.2978	0.3209	0.3583
	GA	10.5280	11.4742	12.4000	2.3348	2.4785	2.7344

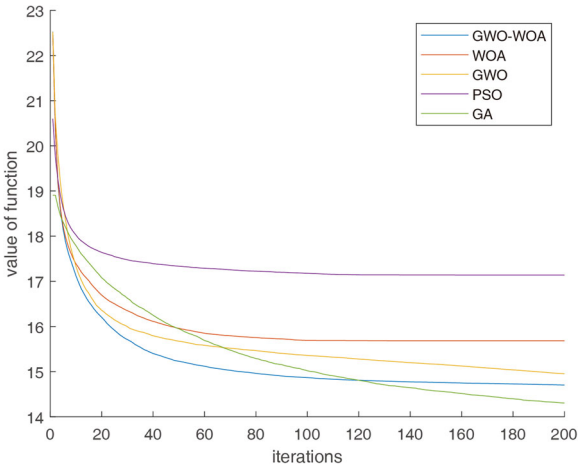

Fig. 1 Convergence curve of algorithms when device number is 60

performance in the processing time. Though WOA has the best processing time, it shows worse performance in the value of the function. The results of both GWO and WOA are worse than those of GWO-WOA. GA has lost its advantage on the value of the function when the device amount is 90.

When the device amount is set to 120, the results obtained by these algorithms are similar to the scenario when the device amount is 90.

Figures 1 and 2 can know that GWO-WOA is the suboptimal algorithm in the perspective of convergence. GA has the best value of the function when the device number is 60. Although the result of GA is better than GWO-WOA, Fig. 1 shows that GWO-WOA converges quicker than GA at the previous iterations. Besides, the message can be obtained that GWO has a quicker convergence speed than WOA. And Fig. 2 shows that GWO is more stable than WOA. As GWO-WOA is improved, it represents reasonable stability, better than the original WOA.

Figure 3 shows that GWO-WOA gets the best result comparing with the other algorithms. Furthermore, the following performance orders are GWO, WOA, GA, and PSO. The advantage of GA has disappeared, as the device number is increased from 60

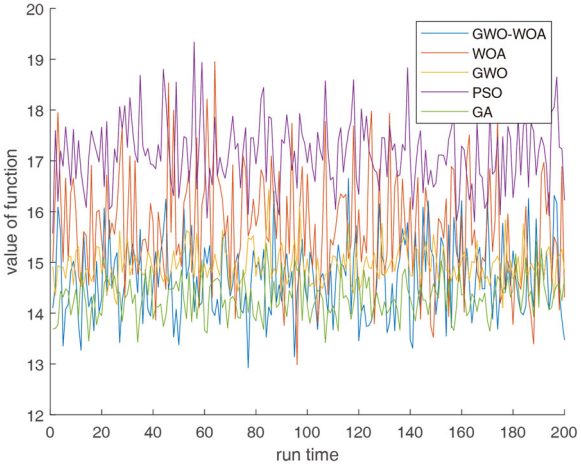

Fig. 2 The results of each run time of algorithms when device number is 60

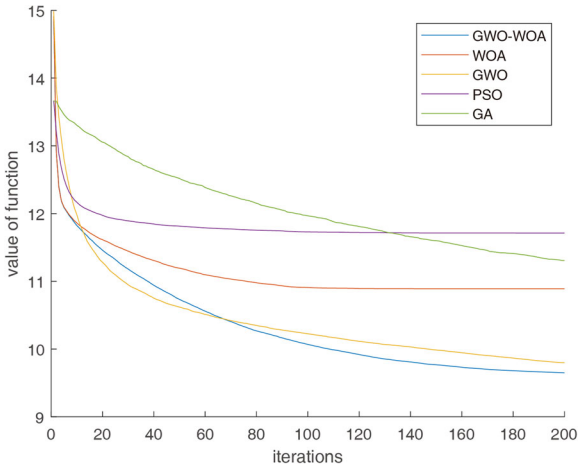

Fig. 3 Convergence curve of algorithms when device number is 90

to 90. The reason may be that the increase in dimensions caused the increase in device numbers. Figure 4 shows that GWO-WOA has good stability and keeps the best results at most times when the algorithms are run. The line of GWO in Fig. 4 shows its stability, which indicates the well performance in avoiding falling into local optima. In comparison, the line of WOA in Fig. 4 fiercely fluctuates, which means it falls into local optima many times.

Figure 5 shows that GWO-WOA takes the place of the best results. While the orders of the function results are the same as Fig. 3 in the situation of the device number is 90, it can conclude that the performance of these algorithms may be sure that even the device number continues to increase, the result will be the same. Figure 6 shows that GWO-WOA has good stability and is the best one in most cases the algorithms run. The stability of GWO is still the best, but we can obtain from Fig. 6 that the gap between GWO-WOA and GWO is reduced. Moreover, in some parts of the figure, the stability of GWO-WOA is better than GWO. It represents that the algorithm improved can well keep the advantage of the original algorithm.

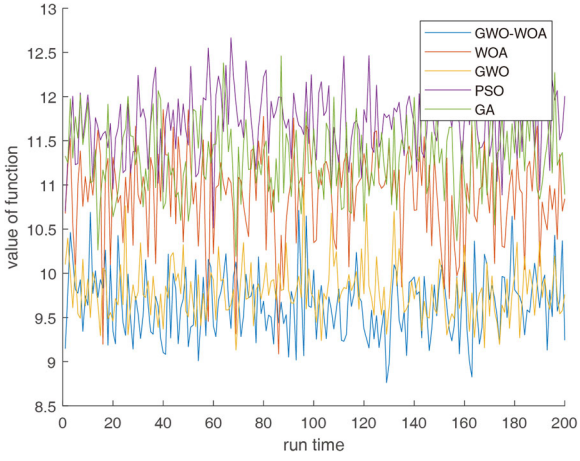

Fig. 4 The results of each run time of algorithms when device number is 90

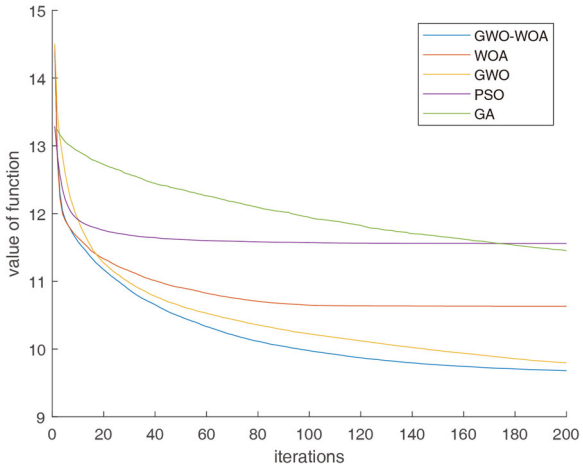

Fig. 5 Convergence curve of algorithms when device number is 120

In general, with the increase of device amount, GWO-WOA performs better than other algorithms in convergence and stability. The algorithm is more suitable for the scenario that with more devices.

6 Conclusions

In this work, we analyzed a computation offloading model with time optimization, energy optimization and price optimization on computation offloading in MEC. Then, normalization is proposed to be used in the model with the purpose of improving the model and eliminating the effects of dimensions. The goal of the model is to get the minimum value. A swarm intelligence algorithm named GWO has been applied to solve the problem. The GWO-WOA algorithm is proposed to search for better solutions for the proposed model. The experiment results show the advantage of GWO-WOA among these algorithms. However, the algorithm proposed still has improvements that can be made. It may not be suitable for the scenario with low dimensions, and its processing time is not the best in the experiments.

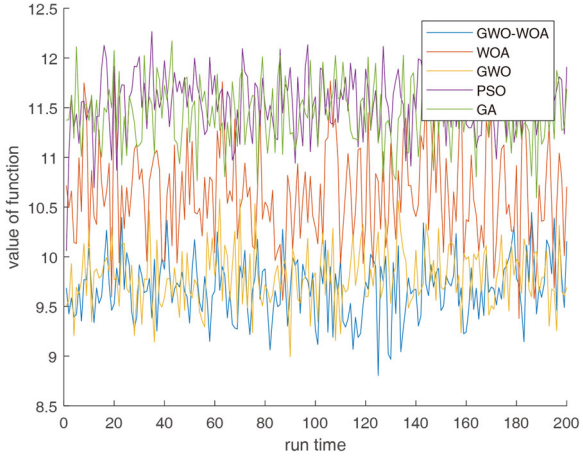

Fig. 6 The results of each run time of algorithms when device number is 120

In future works, we will continue to refine the computation offloading model based on real-world scenarios. Furthermore, we will also investigate how to optimize the offloading strategy by using multi-objective swarm intelligence algorithms and explore more possible methods that can be used.

Abbreviations
IoT: Internet of Things; MEC: Mobile edge computing; GWO: Grey wolf optimizer; WOA: whale optimization algorithm; PSO: Particle swarm optimization; GA: Genetic algorithm; MD: Mobile device; MCC: Mobile cloud computing; 5G: Fifth-generation mobile networks; QoS: Quality of service; NOMA: Non-orthogonal multiple access

Acknowledgements
The authors appreciate help from other colleagues at the Hainan Key Laboratory of Big Data and Smart Services and Hainan Green smart Island Collaborative Innovation Center.

Authors' contributions
Conceptualization: SF, QZ, and YC; data curation: YC; formal analysis: YC; funding acquisition: MH; investigation: YC and QZ; methodology: SF, YC, and QZ; project administration: SF, MH, and FS; resources: YC and QZ; software: YC and QZ; Supervision: SF, MH, and FS; validation: YC; visualization: YC; writing – original draft: YC; writing – review and editing: YC, SF, MH, and FS. All authors have read and agreed to the published version of the manuscript.

Funding
This research was funded by National Key Research and Development Program of China under Grant 2018YFB1404400 and Grant2018YFB1703403 and the Hainan Provincial Natural Science Foundation of China under Grant 2019CXTD400, Hainan Key R&D Program under Grant ZDYF2019115, the National Natural Science Foundation of China under Grant 61865005, the Open Project Program of Wuhan National Laboratory for Optoelectronics under Grant 2020WNLOKF001, Key R&D Project of Hainan province under Grant ZDYF2019020, National Natural Science Foundation of China under Grant 62062030 , and the Education Department of Hainan Province under Grant Hnky2019-22.

Availability of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no competing interest.

Author details
[1]School of Information and Communication Engineering, Hainan University, No. 58 Renmin Avenue, 570228 Haikou, China. [2]School of Sciences, Hainan University, No. 58 Renmin Avenue, 570228 Haikou, China. [3]State Key Laboratory of Marine Resource Utilization in the South China Sea, Hainan University, No. 58 Renmin Avenue, 570228 Haikou, China.

Received: 3 May 2021 Accepted: 21 June 2021
Published online: 08 July 2021

References

1. S. Meng, Y. Wang, Z. Miao, K. Sun, Joint optimization of wireless bandwidth and computing resource in cloudlet-based mobile cloud computing environment. Peer-to-Peer Netw. Appl. **11**, 462–472 (2018). https://doi.org/10.1007/s12083-017-0544-x
2. H. T. Dinh, C. Lee, D. Niyato, P. Wang, A survey of mobile cloud computing: Architecture, applications, and approaches. Wirel. Commun. Mob. Comput. **13**, 1587–1611 (2013). https://doi.org/10.1002/wcm.1203
3. G. Yang, L. Hou, X. He, D. He, S. Chan, M. Guizani, Offloading time optimization via Markov decision process in mobile-edge computing. IEEE Internet Things J. **8**, 2483–2493 (2021). https://doi.org/10.1109/JIOT.2020.3033285
4. H. Zhang, Y. Yang, X. Huang, C. Fang, P. Zhang, Ultra-low latency multi-task offloading in mobile edge computing. IEEE Access. **9**, 32569–32581 (2021). https://doi.org/10.1109/ACCESS.2021.3061105
5. Z. Li, V. Chang, J. Ge, L. Pan, H. Hu, B. Huang, Energy-aware task offloading with deadline constraint in mobile edge computing. EURASIP J. Wirel. Commun. Netw. **2021**, 32569–32581 (2021). https://doi.org/10.1186/s13638-021-01941-3
6. J. Bi, H. Yuan, S. Duanmu, M. Zhou, A. Abusorrah, Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization. IEEE Internet Things J. **8**, 3774–3785 (2021). https://doi.org/10.1109/JIOT.2020.3024223
7. K. Li, Heuristic computation offloading algorithms for mobile users in fog computing. ACM Trans. Embed. Comput. Syst. **20**, 1–28 (2021). https://doi.org/10.1145/3426852
8. Y. Hmimz, T. Chanyour, M. E. Ghmary, M. O. C. Malki, Bi-objective optimization for multi-task offloading in latency and radio resources constrained mobile edge computing networks. Multimed. Tools Appl. **80**, 17129–17166 (2021). https://doi.org/10.1007/s11042-020-09365-9
9. M. N. A. Wahab, S. Nefti-Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms. PLoS ONE. **10**, 1–36 (2015). https://doi.org/10.1371/journal.pone.0122827

10. H. Mazouzi, K. Boussetta, N. Achir, Maximizing mobiles energy saving through tasks optimal offloading placement in two-tier cloud: A theoretical and an experimental study. Comput. Commun. **144**, 132–148 (2019). https://doi.org/10.1016/j.comcom.2019.05.017
11. Y. Zhang, U. Nauman, Deep learning trends driven by temes: A philosophical perspective. IEEE Access. **8**, 196587–196599 (2020). https://doi.org/10.1109/ACCESS.2020.3032143
12. X. Qian, S. Lin, G. Cheng, X. Yao, H. Ren, W. Wang, Object detection in remote sensing images based on improved bounding box regression and multi-level features fusion. Remote Sens. **12**(1) (2020). https://doi.org/10.3390/rs12010143
13. Y. Wu, J. Cao, Q. Li, A. Alsaedi, F. E. Alsaadi, Finite-time synchronization of uncertain coupled switched neural networks under asynchronous switching. Neural Netw. **85**, 128–139 (2017). https://doi.org/10.1016/j.neunet.2016.10.007
14. X. Deng, Z. Sun, D. Li, J. Luo, S. Wan, User-centric computation offloading for edge computing. IEEE Internet Things J. (2021). https://doi.org/10.1109/JIOT.2021.3057694
15. H. G. T. Pham, Q. V. Pham, A. T. Pham, C. T. Nguyen, Joint task offloading and resource management in NOMA-based MEC systems: A swarm intelligence approach. IEEE Access. **8**, 190463–190474 (2020). https://doi.org/10.1109/ACCESS.2020.3031614
16. Q. V. Pham, S. Mirjalili, N. Kumar, M. Alazab, W. J. Hwang, Whale optimization algorithm with applications to resource allocation in wireless networks. IEEE Trans. Veh. Technol. **69**, 4285–4297 (2020). https://doi.org/10.1109/TVT.2020.2973294
17. S. Zhu, W. Xu, L. Fan, K. Wang, G. K. Karagiannidis, A novel cross entropy approach for offloading learning in mobile edge computing. IEEE Wirel. Commun. Lett. **9**, 402–405 (2020). https://doi.org/10.1109/LWC.2019.2957743
18. K. Jiang, H. Zhou, D. Li, X. Liu, S. Xu, in *2020 29th International Conference on Computer Communications and Networks (ICCCN)*, A Q-learning based method for energy-efficient computation offloading in mobile edge computing, (2020), pp. 1–7. https://doi.org/10.1109/ICCCN49398.2020.9209738
19. R. Zhao, X. Wang, J. Xia, L. Fan, Deep reinforcement learning based mobile edge computing for intelligent internet of things. Phys. Commun. **43** (2020). https://doi.org/10.1016/j.phycom.2020.101184
20. Y. Miao, G. Wu, M. Li, A. Ghoneim, M. Al-Rakhami, M. S. Hossain, Intelligent task prediction and computation offloading based on mobile-edge cloud computing. Futur. Gener. Comput. Syst. **102**, 925–931 (2020). https://doi.org/10.1016/j.future.2019.09.035
21. Y. Liu, S. Wang, Q. Zhao, S. Du, A. Zhou, X. Ma, F. Yang, Dependency-aware task scheduling in vehicular edge computing. IEEE Internet Things J. **7**, 4961–4971 (2020). https://doi.org/10.1109/JIOT.2020.2972041
22. M. Huang, Q. Zhai, Y. Chen, S. Feng, F. Shu, Multi-objective whale optimization algorithm for computation offloading optimization in mobile edge computing. Sensors. **21** (2021). https://doi.org/10.3390/s21082628
23. J. Yan, S. Bi, Y. J. Zhang, M. Tao, Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency. IEEE Trans. Wirel. Commun. **19**, 235–250 (2020). https://doi.org/10.1109/TWC.2019.2943563
24. W. Labidi, M. Sarkiss, M. Kamoun, in *2015 22nd International Conference on Telecommunications, ICT 2015*, Energy-optimal resource scheduling and computation offloading in small cell networks (Institute of Electrical and Electronics Engineers Inc., Sydney, Australia, 2015), pp. 313–318. https://doi.org/10.1109/ICT.2015.7124703
25. C. You, K. Huang, H. Chae, B. H. Kim, Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wirel. Commun. **16**, 1397–1411 (2017). https://doi.org/10.1109/TWC.2016.2633522
26. A. P. Miettinen, J. K. Nurminen, in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, Energy efficiency of mobile clients in cloud computing (USENIX Association, Boston, MA, 2010). https://www.usenix.org/conference/hotcloud-10/energy-efficiency-mobile-clients-cloud-computing
27. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014). https://doi.org/10.1016/j.advengsoft.2013.12.007
28. S. Mirjalili, A. Lewis, The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016). https://doi.org/10.1016/j.advengsoft.2016.01.008
29. R. K. Saidala, N. Devarakonda, in *Data Engineering and Intelligent Computing*, Improved whale optimization algorithm case study: Clinical data of anaemic pregnant woman (Springer, Singapore, 2018), pp. 271–281
30. M. Abdel-Basset, D. El-Shahat, I. El-henawy, A. K. Sangaiah, S. H. Ahmed, A novel whale optimization algorithm for cryptanalysis in Merkle-Hellman cryptosystem. **23**, 723–733 (2018). https://doi.org/10.1007/s11036-018-1005-3
31. J. Kennedy, R. Eberhart, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Particle swarm optimization, vol. 4, (1995), pp. 1942–19484. https://doi.org/10.1109/ICNN.1995.488968
32. A. Lambora, K. Gupta, K. Chopra, in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Genetic algorithm- a literature review, (2019), pp. 380–384. https://doi.org/10.1109/COMITCon.2019.8862255

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.