Multimedia Tools and Applications (2023) 82:45453–45469
https://doi.org/10.1007/s11042-023-15490-y

Research on the optimization of energy consumption for multi-priority tasks in mobile computing offloading

Yanhu Zhang[1,2] · **Lijuan Yan**[1]

Received: 3 February 2022 / Revised: 27 April 2022 / Accepted: 18 April 2023 /
Published online: 3 May 2023

Abstract
To solve the problem of insufficient energy consumption for multi-priority tasks in the computing and unloading environments of mobile devices, we designed evaluation methods to improve the traditional simulated annealing algorithm and obtain the optimal allocation scheme for these tasks under multiple computing resources. Firstly, we proposed the task model and discussed the classification and definition of task priority as well as the procedure of task processing in detail. Secondly, a calculation model for the energy consumption of mobile devices is provided. Thirdly, an evaluation mechanism is designed to evaluate the unloading distribution scheme of mobile devices effectively. Finally, the improved traditional simulated annealing algorithm was applied to find the optimal distribution scheme for this computing environment. All allocation schemes obtained in this study were compared and analyzed via simulation. The results showed that the proposed algorithm can reduce the energy consumption of mobile devices more significantly, shorten the system response time, and complete high-priority tasks based on time constraints.

Keywords Mobile edge computing · Computation offloading · Multiple priority tasks · Simulated annealing algorithm · Evaluation value · Energy consumption model · Energy efficient

1 Introduction

As the mobile Internet continues to develop, intelligent devices have produced a tremendous amount of data. According to the literature [4, 10, 19], the current demand for smart mobile devices suggests that global data from the mobile devices will overwhelm the available network capability, thereby adversely affecting the user's experience of new technologies (e.g. the Internet of Things and VR). In order to alleviate such a contradiction, the

✉ Lijuan Yan
juanjanny@qq.com

Yanhu Zhang
forzyh@163.com

[1] Guangdong Songshan Polytechnic, Shaoguan 512126, Guangdong, China

[2] Jose Rizal University, 1550 Mandaluyong, Metro Manila, Philippines

concept of Mobile Edge Computing (MEC) or cloud computing was proposed. The core idea of MEC is to place the server that is capable to process computing powers close to the mobile terminal and send some data originally planned for the cloud server to the edge server for processing. In this way, the network data transmission load could be reduced, which could improve the system's overall performance.

Scholars have studied the energy consumption optimization of mobile devices from various directions, and the findings have been abundant. Specifically, Yu Bowen et al. [1] proposed a task unloading algorithm based on the COMED framework to optimize the overall energy consumption of base stations and devices. Zha et al. [24] designed a task unloading algorithm based on energy efficiency and used the auction theory to find the optimal task unloading solution. Sivanandam et al. [18] proposed a multiprocessor scheduling algorithm based on particle swarm optimization and transformed the particle swarm vector into a scheduling-first model. Meng et al. [14] provided a random task transfer algorithm based on machine learning that could generate the optimal transfer strategy for random tasks when combined with the improved Q learning and deep learning algorithms. Liu et al. [13] studied the deep learning task offloading and proposed a set of sparse beam forming frameworks. [22] Designed a task scheduling algorithm to minimize energy consumption with particle swarm optimization of multiple resources in the edge terminals. Zhou et al. [28] proposed an improved game theory algorithm to reduce the task computation time. Goyal S et al. [6] proposed a cloud-based optimized framework for energy-resource allocation based on the Whale Optimization Algorithm to minimize energy consumption. However, these researches are primarily focused on optimizing the allocation of computing resources with optimization algorithms to reduce the energy consumption of mobile devices. They failed to address the energy consumption of mobile devices within the scenario of multiple computing resources.

Another direction to optimize energy consumption is to prioritize the satisfaction of time constraints. Specifically, Xu et al. [9] proposed a computational unloading model to reduce the energy consumption of mobile terminals while meeting the time constraints. Yang et al. [23] considered the capacity constraints of lead-time, backhaul links, and the maximum user delay and proposed an effective unloading scheme to minimize the total network energy consumption. Zhang et al. [26] adopted an artificial fish swarm algorithm to design the offloading strategy of energy consumption optimization with time delay constraints. Zhang et al. [27] proposed a computation offloading scheme for energy perception by weighing energy consumption and time delay and introduced the residual battery energy of the smart devices into the definition of the weight factor, thereby reducing the total system energy consumption remarkably. These studies adopted strategies to reduce the energy consumption of mobile devices while satisfying the task time constraint. No in-depth discussions were performed on optimizing energy consumption in multi-resource scenarios.

Moreover, scholars have investigated the optimization of equipment energy consumption under multiple computing resource scenarios. Particularly, a reasonable optimization strategy for energy consumption of multi-equipment calculation unloading has been provided and discussed in detail [2, 3]. Kim Y et al. [11] calculated the discharge and scheduling of mobile edge server resources to optimize the energy consumption and efficiency for mobile devices and the server. Tong et al. [20] proposed a task offloading and resource allocation algorithm in the MEC environment. Ding et al. [5] studied the code-oriented partition to compute the offload strategy and determine the user's execution locations and minimized the system overhead with an offload strategy, but the authors failed to take task parallelism into account. Li et al. [12] proposed a switching strategy to calculate discharge so that the mobility time caused by increased unloading

may be shortened. Zhao et al. [21] proposed a privacy perception computing offloading algorithm based on the Lyapunov optimization theory. Similar work had been performed in other reports as well [7, 8, 15–17, 25]. In these studies, the authors investigated the optimization of energy consumption of mobile devices under the scenario of multiple computing resources but didn't discuss the optimization problem of energy consumption in the case of multi-priority tasks.

Despite the promising and abundant results in optimizing the energy consumption of mobile devices in a multi-task environment, little research has addressed the energy consumption optimization problem for mobile devices in a multi-priority task environment. In practice, tasks may be categorized as urgent, important, and general tasks based on their urgency. Unwanted consequences would emerge if all such tasks are treated in the same way. Therefore, during MEC task processing, task priority must be considered.

In the literature, a study similar to this one was performed by Zhu et al. [29], in which the task priority was divided into five levels. The authors used the auction algorithm to process tasks according to their priorities, and the obtained task allocation was satisfactory. Even so, the authors didn't consider the sequence of task entry and the waiting time after admission, leading to possible prolonged waiting for some tasks.

Zhou et al. [28] developed a new cooperative unloading mechanism to improve the experience of mobile users and specifically investigated the processing of multi-priority tasks. In order to reduce the devices' energy consumption, the author suggested establishing a task center to manage mobile users more effectively and achieve low-delay communication. Besides, the tasks could be pre-processed according to their task priority to enhance the efficiency of task input and improve user experience. And finally, the Double DDPG algorithm was proposed to ensure the lowest service delay.

Task priority allocation in previous literature has been primarily based on the time constraints without considering the situation in which the task priority changes during task assignments or under the waitlist for execution. In some cases, such changes may result in late processing of the urgent tasks or a long waiting time for low-priority tasks.

To address such a problem, both the sequence of task admission and the waiting time after task assignment are taken into account in this study. An appropriate algorithm to mark task priorities according to the waiting time is proposed, thereby addressing the problem of prolonged waiting and optimizing the energy consumption of mobile devices in the case of multi-priority tasks.

Besides, in order to solve the problems mentioned above, a computational unloading method (MPT algorithm for short) for multi-priority tasks was developed to optimize the energy consumption of mobile intelligent devices under the environment of multi-computing resources. With the MPT algorithm, the prolonged waiting time for both high-priority and low-priority tasks may be addressed, providing insights for scenarios with similar requirements. The MPT algorithm could be summarized in 3 steps. Firstly, a method to quantify the priority of computing tasks was designed to assign a priority value to each computing task. Secondly, a strategy to quantify and evaluate the energy consumption of unloading schemes was designed. And finally, an improved simulated annealing algorithm was proposed to optimize the computational unloading scheme based on the evaluation strategy. With the improved algorithm, a computational unloading scheme that is capable to reduce the energy consumption of mobile devices was derived.

Since the traditional simulated annealing algorithm is insufficient to optimize the unloading scheme under multi-computing resources, it was improved in this paper according to the current scene to optimize the unloading scheme and minimize the energy consumption of mobile devices.

Our major contributions are summarized as follows. Firstly, an appropriate solution for multi-priority tasks in a multi-computing resource environment is proposed. Secondly, a comprehensive evaluation strategy for energy consumption and aging performance is provided for tasks with different priorities. Finally a promising algorithm to optimize energy consumption for multi-priority tasks in the multi-resource moving-edge computing-unloading environment is given, which produces the final computing unloading execution scheme with the best overall performance.

The rest of this paper is organized as follows. Section 2 explains the concepts involved in this paper; Section 3 gives the calculation workflow for the task priority values under a multi-priority task environment. Section 4 gives the energy consumption calculation model for each resource under the multi-resource environment; Section 5 presents the calculation workflow and demonstrates the evaluation mechanism as well as the optimization algorithm for the total energy consumption of mobile devices; Section 6 uses a simulation experiment to test the proposed algorithm; Section 7 concludes this paper.

2 Task processing model

2.1 Task priority

Tasks on the to-do list differ by their urgency. Some tasks are so urgent that they must be dealt with immediately while other tasks may wait for later processing. The urgency levels usually include urgent (requiring immediate computing), important (requiring computing as soon as possible), and normal (may wait for later computing).

If all tasks are sorted randomly for calculation and unloading, urgent and important tasks are probably not handled in time, leading to possible unwanted consequences. Therefore, a strategy is proposed to divide the tasks into several priority levels according to emergency levels so that the tasks can be unloaded reasonably in mobile edge computing (Fig. 1).

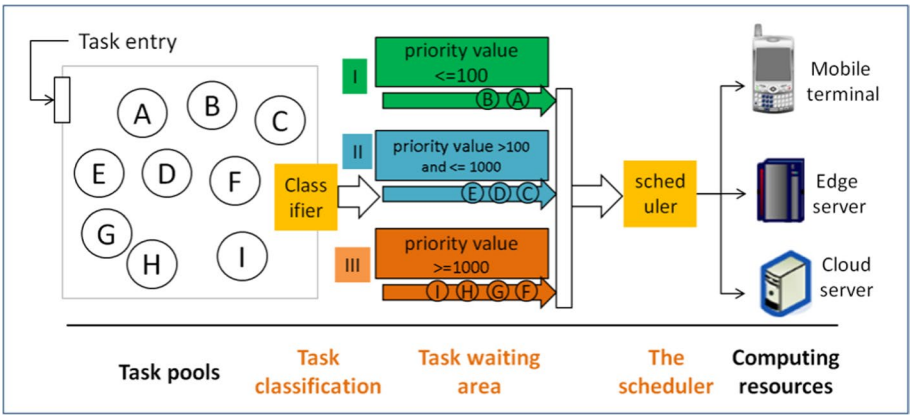


Fig. 1 Multi-priority task processing flow chart

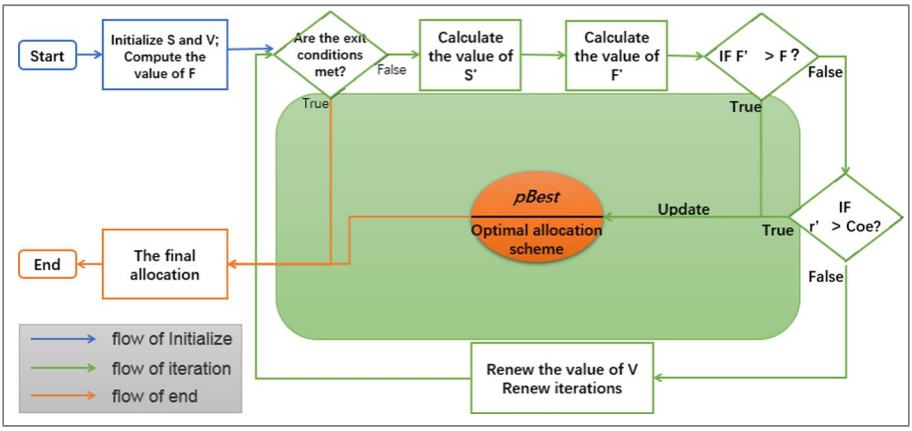


Fig. 2 The flow chart of the MPC algorithm

Table 1 Task-related data

Task no.	A	B	C	D	E	F	G	H	I
load	1200	200	3200	1400	3000	500	800	720	1450
The amount of data	2600	1800	1600	700	500	220	140	65	150
priority	I	I	II	II	II	III	III	III	III
Order no.	5	6	3	4	9	1	2	7	8
The priority value	15	16	103	104	109	1001	1002	1007	1008

2.2 Modeling the task processing flow

The execution process for priority tasks in the workflow system is given in Fig. 2. The tasks in the task pool, in which the pending tasks are stored, are mutually independent, and the corresponding parameters of each task are shown in Table 1.

The task pool has only an entry and an exit. When the task entry portal is open, new tasks can enter the task pool. The classifier selects the pending tasks and allocates these tasks to different pending areas based on certain rules. Each pending area has three regions (I, II, and III) with different priority levels. Region I has the highest priority and preempts the resources of any other non-executing tasks. Region II can preempt all resources in Region III and Region III has the lowest priority for resource allocation.

When no tasks are within the task pool and the time area is fixed at every interval, the task pool entry opens. When the number of tasks in the task pool exceeds the predetermined limit, no new tasks shall be received. Since the task entry in the task pool is opened for a pre-defined time, new and old tasks will be mixed together in the task pool. At this time, the classifier determines the task sequence by the priority value T of each task, which is calculated by the equation below:

$$\text{Priority value T} = 10^{\text{priority}} + \text{order value} \tag{1}$$

The tasks in the waiting area shall be executed in the sequence of Regions I, II, and III, and the tasks in the same pending area shall not be prioritized for optimal energy consumption.

In the task pool, each task is calculated a fixed number of times and assigned to a computing resource with its data. These data are transmitted to the corresponding operations through the network, and each task has a priority rank, in which the value I represents the highest priority, followed by II and III.

As the tasks enter the task pool, the system assigns an admission number and a priority value T for each task according to their entry sequence and Eq. 5. Subsequently, the classifier distributes the tasks to the corresponding task waiting area based on their T values, and the scheduling optimization algorithm is executed by the scheduler to assign computing resources to these tasks. In some cases, the system takes too long to finish these assignments due to a large amount of computation in Regions I and II, so the tasks in the task pool would have their priorities changed. Under such a situation, the original assignment would be unfair to tasks with changed priorities, and the time threshold TPLIM is set in the system to evade such unfairness. When the total time to run tasks in Regions I and II exceed TPLIM, the classifier redeploys the tasks and loads them into the three Regions according to the latest task priority.

This article is primarily focused on the situation in which the task priority can change dynamically. In some special scenarios, tasks constantly enter the high-priority task pool. If such entry is continuous, tasks with lower priority values might wait for a long time before being performed, which is particularly unfair for such tasks and unwanted for good user experience. Therefore, Formula 1 was proposed to reflect the overall dynamic change of task priority, in which multiple factors such as the emergency states and the task waiting time are taken into consideration.

When the classifier detects that all tasks in the area to be processed are finished, the tasks in the task pool will be sorted according to their priority values. Those tasks with a priority value smaller than *m1* will be assigned to Region I to wait. Regions II and III would receive tasks with a priority value within *m1- m2* and greater than *m2*, respectively($m1 < m2$). Subsequently, the scheduler allocates computing resources in the order of I, II, and III. For tasks in the same priority level, their priority values are considered identical when dispatching computing resources to optimize the energy consumption of mobile devices.

3 Calculation models

3.1 Total energy consumption model of mobile devices

In the SA algorithm, an evaluation value is used to evaluate an allocation scheme. In a multi-task and multi-resource calculation unloading model, the evaluation value is also used to evaluate the resource allocation scheme comprehensively to achieve the lowest energy consumption.

When calculating the total energy consumption of a device, assume that $T = \{T1, T2, T3\}$ is an existing calculation unloading scheme, where $T1 = \{t_1, t_2 \ldots t_n\}$ is a set of n tasks assigned to mobile edge devices, $T2 = \{t_{n+1}, t_{n+2} \ldots t_m\}$ is a set of m-n tasks assigned to the edge server, and $T3 = \{t_{m+1}, t_{m+2} \ldots t_k\}$ is a set of k-m tasks assigned to the cloud server. The total energy consumption E_{sum} generated by the mobile devices when T is completed is:

$$E_{sum} = E_{T1} + E_{T2} + E_{T3} + E_{dst} \tag{2}$$

Where E_{sum} is the total energy consumption of the mobile devices; E_{T1} is the energy consumption of the mobile edge devices to complete *T1;* E_{T2} is the energy consumption of the mobile edge devices when the edge server completes *T2*; E_{T3} is the energy consumption of the mobile edge devices when the cloud server completes *T3*; E_{dst} refers to the energy consumption when running the distribution algorithm, which is calculated as:

$$E_{dst} = P_{run} * \frac{C_{codeNum}}{S_{edge}}^{*} \lambda \tag{3}$$

Where P_{run} is the running power of the mobile edge device, $C_{codeNum}$ is the total number of instructions in the distribution algorithm (calculated according to the maximum number of iterations in the corresponding algorithm or the number of empirical iterations), S_{edge} is the running speed of the mobile edge device, and λ is an adjustment coefficient.

3.2 The calculation model for evaluation values

Two preconditions exist in this paper when reducing the energy consumption of mobile devices. Firstly, the time required for the system to complete the total task should meet the time constraint T_{lim}. Secondly, tasks with higher priority have tighter time constraints than their counterparts with lower priority. Assume λ_1、λ_2、λ_3 are the time constraint coefficients for priority levels I, II, and III, respectively. In other words, tasks of priority I, II, and III shall be completed within $\lambda_1 * T_{lim}$, $\lambda_2 * T_{lim}$, and $\lambda_3 * T_{lim}$, respectively, or their evaluation values will be penalized. Following that, the numbers of tasks that are not completed within their corresponding time constraints in the allocation scheme are counted separately by priorities. Assume that the total number of tasks of priority I is Ci and the number of unfinished tasks is CN_i, then we have the following equation to obtain the value for evaluation:

$$F = \begin{cases} E_{sum} & T_{sum} \le T_{\lim} \; and \; CN_i == 0 \\ E_{sum} * \frac{T_{sum}}{T_{\lim}} * \alpha & T_{sum} > T_{\lim} \; and \; CN_i == 0 \\ E_{sum} * \frac{C_i}{C_i - CN_i} * \beta & T_{sum} \le T_{\lim} \; and \; CN_i <> 0 \\ E_{sum} * (\frac{T_{sum}}{T_{\lim}} * \alpha + \frac{C_i}{C_i - CN_i} * \beta) & T_{sum} > T_{\lim} \; and \; CN_i <> 0 \end{cases} \tag{4}$$

Where, E_{sum} is the total energy consumed by mobile edge devices; T_{sum} is the maximum time consumed by all equipment participating in computing the force unloading task; T_{lim} is the time constraint; α and β are the total and priority task time constraint adjustment factors, respectively.

The calculation equations for E_{sum} and T_{sum} are:

$$E_{sum} = E_{edge} + E_{ec} + E_{cloud} \tag{5}$$

$$T_{sum} = \text{Max}\left(\; T_{edge}, \; T_{ec}, \; T_{cloud} \; \right) \tag{6}$$

C_i and CN_i are calculated by the following equations:

$$C_i = \sum_{j=1}^{n} \left(C_{i-1} + 1\right) C_i \; priority \; = i \tag{7}$$

$$CN_i = \sum_{j=1}^{n} \left(CN_{i-1} + 1\right) C_i \; priority \; = i \; And \; \; TN_i \; < \; \lambda_1^* T_{lin} \tag{8}$$

4 Improved algorithm

4.1 Algorithm flow chart

Where S represents the position of the allocation scheme, S 'represents the position of the new allocation scheme, V represents the speed, F is the evaluation value of the allocation scheme, r' is a random value between *0* and *1*, and C_{oe} is the set threshold. As the number of iterations increases, the F value gradually approaches 0.

4.2 Implementation of algorithm

The traditional SA algorithm compares the evaluation values F of the new and current positions. When the new position has a bigger evaluation value than the current position, the allocation scheme represented by the new position is considered better and the new position is accepted. Otherwise, the allocation scheme of the new position is considered inferior to the current one.

In order to exclude the possibility of the new evaluation value being the local optimum, the system is designed to accept the new position by comparing the random number r with the variable factor, and the variable factor's value tends to be 0. Even with the ability to solve the local optimum problem, the traditional simulation algorithm is still unsatisfactory because in most cases, the resulting resource allocation scheme by the end of the algorithm is not the optimal solution.

Therefore, in this study, we propose an improved SA algorithm based on mobile edge calculation unloading in the multi-resource environment to solve the local optimum problem and obtain the optimal solution. Specifically, an attribute is added to the annealing factor to store the allocation scheme *pBest* corresponding to the optimal evaluation value of the annealing factor. When the annealing factor fails to obtain a solution better than *pBest* and the consecutive *Res* or the total execution time for level I and II tasks exceed TP_{lim} in the search process, the iteration ends and the optimal solution *pBest* is obtained. The algorithm implementation details are described below:

5 Experiment and result analysis

5.1 Experimental environment

Matlab 2016A was used to perform the simulation experiment. Intel(R) Xeon dual-core 2.4G CPU and 4G memory 4G were the hardware configuration used in the simulation. The time constraints include 1) no more than $1/3*V_{ave}$ for priority I tasks; 2) no more than

Input: algorithm iteration times Ite, task Tasks, time constraint T_{lim}, time constraint TP_{lim}, cloud server CSs, edge device Edges, Edge server SEs, iteration exit coefficient Res (in successive Res iterations, no new optimal allocation scheme is obtained).
Output: Priority task resource allocation scheme $S_{eng\text{-}low}$

```
For q=1 to 3        // Perform tasks in the three priority areas in sequence
  If T_pre < TP_lim   // Check whether the running times of Region I and II exceed TP_lim
      Read tasks in area q;
      Initialize the task assignment and resource allocation scheme S, search speed V;
      Calculate the energy and time consumption of mobile edge equipment when unloading task sets for three computing force resources according to S.
      Calculate the total energy consumption, total time consumption, and the corresponding evaluation value for distribution scheme S;
      Initialize the current optimal allocation scheme pBest_p;
      For i=1 to Ite do
          Update allocation scheme S_i according to the latest speed;
          Calculate the energy consumption and time consumption of mobile edge devices when tasks are unloaded to three kinds of resources based on S_i.
          Calculate the total energy consumption, total time consumption, and the corresponding evaluation value for S_i:
            If the energy consumption of allocation scheme S_i is less than that of pBest_p
                Update pBest_p to the energy consumption value of the new scheme;
            Else
                The system generates a random number r between 0 and 1
                If r < C_oe
                  Update pBest_p as the latest energy consumption value of the scheme;
                End if
            End if
            If this iteration finds a better task allocation scheme
                timer = 0;
            Else
                timer = timer + 1;
            End if
            Update search speed V_i;
            If timer > Res
                Exit the iteration and get the optimal allocation scheme pBest_p;
            End
            End for // End for the current iteration and proceed to the next iteration
        Else
            Break; // Jump out of the loop and run the task sorter to reload the task
        End if          //
    Return pBest_p; // Returns the optimal allocation scheme for the current q block
    Collect statistics on the total power consumption and total time consumption of mobile edge devices
End for             // End task scheduling for all areas
```

Algorithm 1 Energy consumption optimization algorithm for priority tasks in a multi-resource mobile edge computing unloading environment

2/3*V_{ave} for priority II tasks;.3) no more than 1.5*V_{ave} for priority III tasks. Particularly, V_{ave} is the average speed of processing computing resources for all participating tasks. The detailed parameters of the proposed algorithm are shown in Table 2.

5.2 Experimental results and analysis

The algorithm in this paper is analyzed and verified by comparing its unloading strategies with those of other algorithms.

5.2.1 Comparison of unloading strategies

The unloading strategy obtained by the proposed algorithm was compared with 5 other strategies from the dimensions of energy consumption evaluation and mobile device operation times to verify the algorithm's comprehensive performance. The so-called other strategies for comparison are derived from the traditional SA algorithm, namely no unloading at all (MBC), complete unloading to edge server (ESC), complete unloading to cloud server (CSC), Ref. [28], and mixed unloading strategy (SA).

Table 2 Detailed parameters of the proposed algorithms

Description	Value
size of data	10–350 MB
amount of calculation	100–3500 CPU Cycles
running power of mobile devices	60 MW
sending power of mobile devices	15 MW
receiving power of mobile devices	5 MW
standby power of mobile devices	3 MW
computation capacity of local devices	0.5 GHz
computation capacity of edge devices	1.7 GHz
computation capacity of cloud servers	2.5 GHz
speeds of uploading data to edge servers	12 MB/s
speeds of downloading data from edge servers	24 MB/s
speeds of uploading data to cloud servers	7.2 MB/s
speeds of downloading data from cloud servers	14.4 MB/s
workflows	100–500 tasks
channel bandwidth	$5.0*10^{-3}$GHz
background noise power	$1.0*10^{-13}$w

Comparison of energy consumption In this study, the evaluation values for the energy consumption of mobile devices were compared in 5 unloading schemes, where the numbers of tasks were 100, 200, 300, 400, and 500, respectively.

The SA, Ref. [28], and the proposed algorithms were tested 500 times, and the maximum iteration number in each test was set to 1000. The optimal energy and time consumption results were obtained for each distribution scheme, and the average results of the 500 experiments were taken for analysis. The hybrid unloading strategy was implemented to find better resource allocation schemes in the experimental environment, and the proposed algorithm was verified to show better performance than the traditional SA algorithm. The specific energy consumption evaluation values of the algorithms tested are shown in Fig. 3.

Comparison of running time The times required by the six unloading strategies mentioned above are compared (Fig. 4). When the task volume was 200, the running time of the proposed algorithm is 2.34 times, 2.18 times, 3.12 times, and 1.33 times shorter than the MBC, the ESC, the CSC, the Ref [28], and the SA strategies, respectively. Therefore, the unloading strategy obtained with the proposed algorithm can reduce the system response time remarkably, and it also shows that the complexity of MPT algorithm is better than other algorithms.

Comparison of computational complexity A brief comparison of the time complexity of the above algorithms was provided, and the result shows that the algorithm complexity of MBC, ESC and CSC is $O(1)$, and the algorithm complexity of Ref. [28], SA and MPT is $O(n)$.

5.2.2 Comparison of reference algorithms

The algorithms mentioned in references [2, 3, 9, 28], and were compared with the proposed algorithm on a number of important indicators. The experimental parameters are similar to those in 5.2.1, and each algorithm was compared with 500 experiments and the maximum

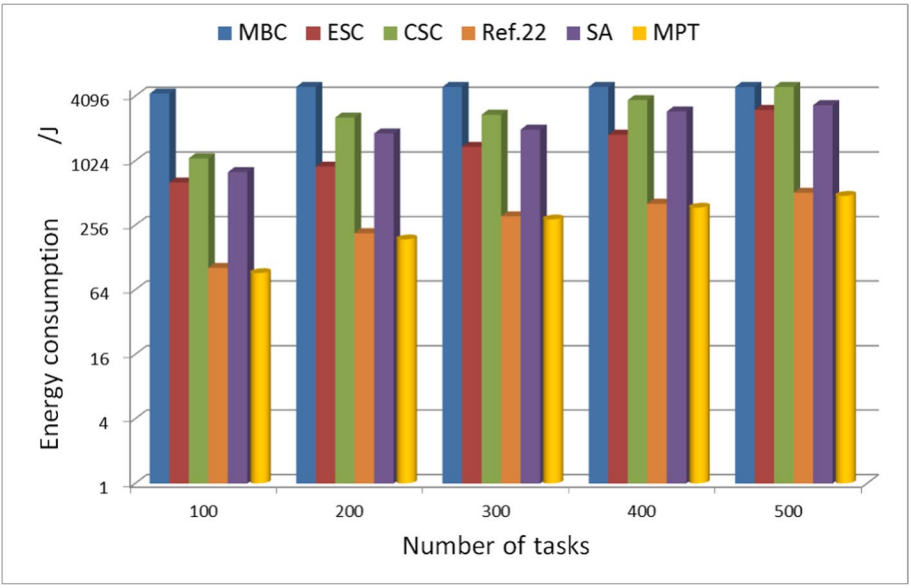


Fig. 3 Comparison of energy consumption of each unloading strategy

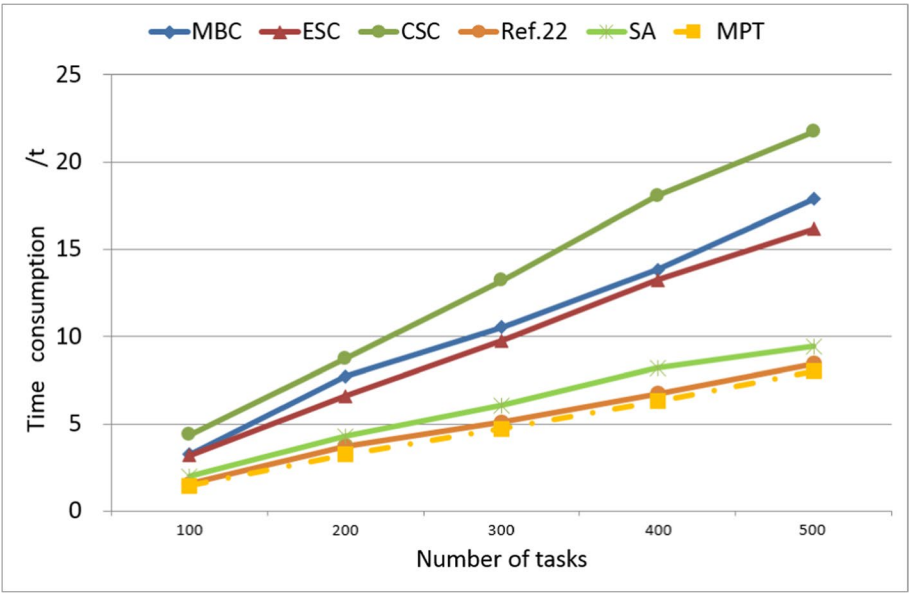


Fig. 4 Response time comparison of various unloading strategies

iteration number of 1000. The average value of the 1000 iterations was taken as the data for comparison.

Comparison of energy consumption The energy consumption of mobile devices over the last distribution scheme (500 tasks) was calculated by each algorithm for comparison (Fig. 5). The energy consumption evaluation value of the proposed algorithm is identified to be superior to that in references [2, 3, 28], and, but inferior to that in reference [9].

Convergence performance comparison The proposed algorithm was compared with the traditional SA algorithm as well as the algorithms in references [2, 3, 9, 28], and to verify its convergence. When the algorithm ended, the number of iterations corresponding to the scheme was recorded. Each algorithm ran for 500 times, and the average iteration numbers were calculated for comparative analysis (Fig. 6). The proposed algorithm is superior to the traditional SA algorithm as well as the algorithms mentioned in references [2, 3, 28], and but inferior to the algorithm in reference [9]. Nevertheless, compared to the algorithm in reference [9], the proposed algorithm still exhibits higher overall stability.

Comparison of task completion rates According to the parameter settings in 5.1, the time constraints of high-priority tasks are more stringent. In order to reveal the actual situations in which tasks with varied priorities are completed within the time constraints, the allocation schemes of the six unloading strategies were run 500 times to calculate the average completion rates (Fig. 7).

With the proposed algorithm, the on-time completion rate for Level I tasks is over 100%. For Level II and Level III tasks, the on-time completion rates are over 98%. All on-time completion rates are significantly higher than those obtained from other algorithms.

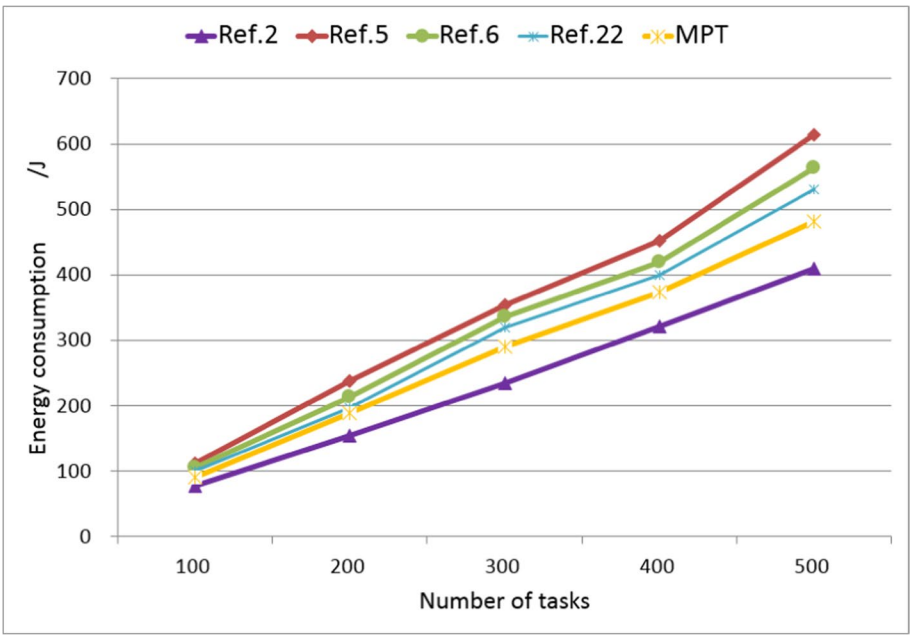


Fig. 5 Comparison of energy consumption across reference algorithms

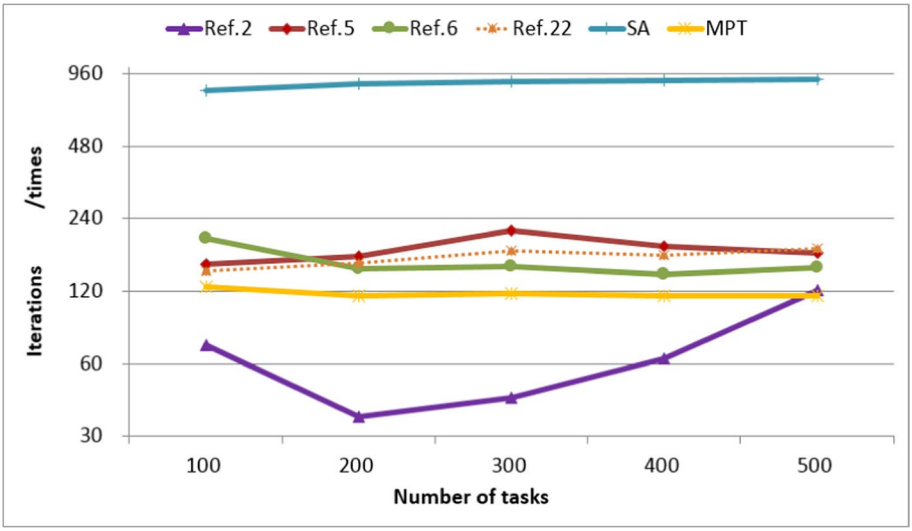


Fig. 6 Comparison of convergence performance

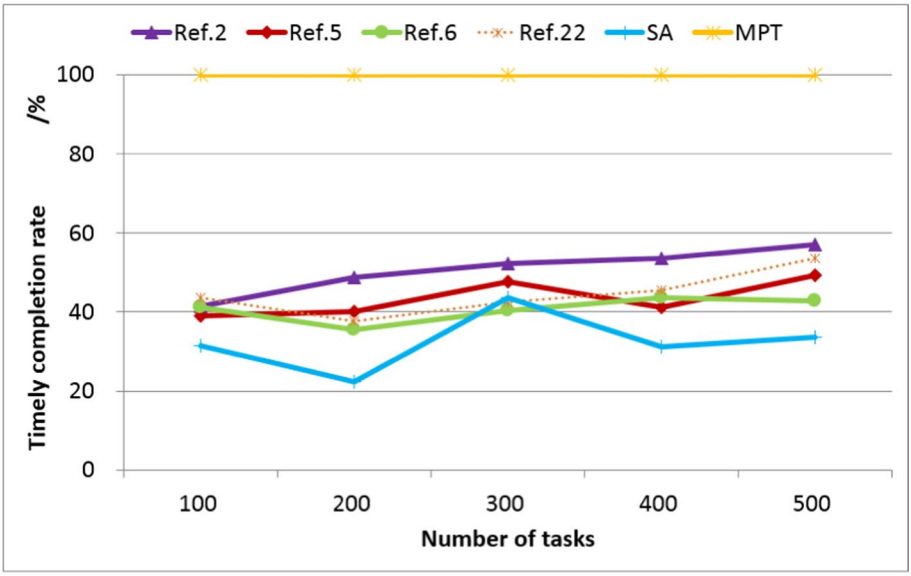
We recommend an algorithm that is suitable for optimizing the energy consumption of mobile devices in multi-priority task and multi-resource environment. This algorithm can ensure that the task with high priority can be executed first. In addition, the recommendation algorithm considers the waiting time of all tasks in a balanced way when determining the execution order. Compared with other algorithms, simulation experiments show that MPT algorithm can better reduce the energy consumption of mobile devices and prolong the life cycle of mobile devices on the basis of ensuring that high-priority tasks are processed in time.

6 Open issues

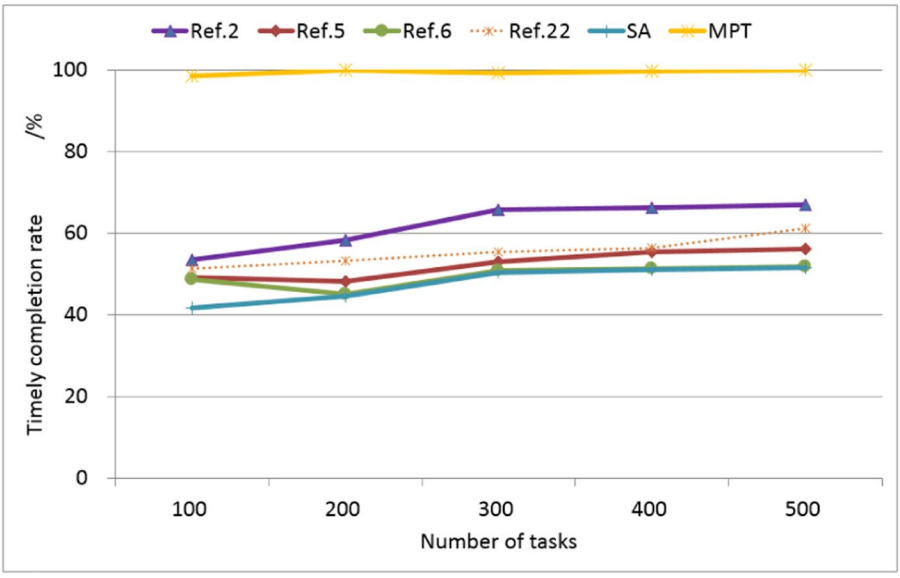
In this paper, when dealing with the priority of tasks, it is assumed that the tasks to be dealt with are relatively fixed in a certain time segment. In actual situations, the number and type of tasks are likely to change dynamically, which is not considered in this paper and will be the direction of research in the next stage.

7 Conclusion

In order to improve the battery life of mobile devices under multiple priority tasks, this paper proposed a mobile edge equipment energy consumption calculation model, which was used to calculate the energy consumption of equipment under the scenario of multi-priority tasks. Subsequently, we designed a rating scheme that can be used for evaluation. and finally, an improved SA algorithm is implemented to find the optimal solution. Simulation results show that the proposed algorithm can complete almost all tasks of all priorities within the constraint time, and the common tasks meet the response time. The proposed



a Time-bound completion rates for Level I tasks



b Time-bound completion rates for Level II and Level III tasks

Fig. 7 Time completion rates for tasks with differed priorities. **a**. Time-bound completion rates for Level I tasks. **b** Time-bound completion rates for Level II and Level III tasks

task scheduling scheme can reduce the energy consumption of mobile devices remarkably with satisfactory convergence.

Funding This work was supported by the Characteristic Innovation Research Fund for Universities of Guangdong Province (No.2019GKTSCX041) and the Science and Technology Program of Shaoguan (No. 210722094530279).

Declarations

Competing financial interests The authors declare no competing financial interests.

Conflict of interest The authors declare no conflict of interest.

References

1. Bowen Y, Lingjun P, Yuting X et al (2018) Joint task offloading and base station association in mobile edge computing[J]. J Comput Res Dev 55(3):537–550
2. Changsheng Y, Kaibin H et al (2017) Energy-efficient resource allocation for mobile-edge computation offloading[J]. IEEE trans on. Wirel Commun 16(3):1397–1411
3. Chen X, Lie J, Wenzhong L et al (2016) Efficient multi-user computation offloading for mobile-edge cloud computing. [J]IEEE/ACM Trans on Netw 24(5):2795–2808
4. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update (2016) 2016—2021 White Paper[EB/OL].(2016-03-05)
5. Ding Y, Liu C, Zhou X, Liu Z, Tang Z (2019) A code-oriented partitioning computation offloading strategy for multiple users and multiple Mobile edge computing servers. IEEE trans. Ind. Inform 16:4800–4810 [CrossRef]
6. Goyal S, Bhushan S, Kumar Y, Rana AuHS, Bhutta MR, Ijaz MF, Son Y (2021) An optimized framework for energy-resource allocation in a cloud environment based on the whale optimization algorithm. Sensors 21(5):1583. https://doi.org/10.3390/s21051583
7. Gupta D, Rani S, Ahmed SH, Verma S, Ijaz MF, Shafi J (2021) Edge caching based on collaborative filtering for heterogeneous ICN-IoT applications. Sens 21:5491. https://doi.org/10.3390/s21165491
8. He X, Lu H, Huang H et al (2020) QoE-Based cooperative task offloading with deep reinforcement learning in mobile edge networks. IEEE Wirel Commun 99:2–8. https://doi.org/10.1109/MWC.001.1900406
9. Jia XU, Xaeiun LI, Ruimiao D et al (2019) Energy efficient multi resource computation offloading strategy in mobile edge computing[J]. Comput Integr Manufact Syst 25(4):954–961. https://doi.org/10.13196/j.cims.2019.04.018
10. Jiena L, Jia-bo Z, Zufan Z et al (2020) Survey of mobile edge computing ofloading strategies [J]. J Chin Comput Syst 41(9):1866–1877
11. Kim Y, Kwak J, Chong S (2015) Dual-side dynamic controls for cost minimization in mobile cloud computing systems [C]// Proc of the 13th IEEE Int Symp on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks. Piscataway, NJ:IEEE, :443–450
12. Li B, Niu L, Huang X, Ding H (2020) Mobility prediction based computation offloading handoff strategy for vehicular edge computing. J Electron Inf Technol 42(11):2664–2670. https://doi.org/10.11999/JEIT190483
13. Liu L, Liu X, Zeng S, Wang T, Pang R (2019) Research on virtual machines migration strategy based on mobile user mobility in mobile edge computing. J Chongqing Univ Posts Telecommun 31:158–165
14. Meng H, Huo R, Guo Q, Huang T, Liu Y (2019) Machine learning-based stochastic task offloading algorithm in mobile-edge computing[J]. Journal of Beijing University of Posts and Telecom 42(2):25–30. https://doi.org/10.13190/j.jbupt.2018-078
15. Panigrahi R, Borah S, Bhoi AK, Ijaz MF, Pramanik M, Kumar Y, Jhaveri RH (2021) A consolidated decision tree-based intrusion detection system for binary and multiclass imbalanced datasets. Math 9:751. https://doi.org/10.3390/math9070751

16. Panigrahi R, Borah S, Bhoi AK, Ijaz MF, Pramanik M, Jhaveri RH, Chowdhary CL (2021) Performance assessment of supervised classifiers for designing intrusion detection systems: a comprehensive review and recommendations for future research. Math 9:690. https://doi.org/10.3390/math9060690
17. Rani S, Koundal DK, Ijaz MF, Elhoseny M, Alghamdi MI (2021) An optimized framework for WSN routing in the context of industry 4.0. Sens 21:6474. https://doi.org/10.3390/s21196474
18. Sivanandam SN, Visalakshi P, Bhuvaneswari A (2007) Multiprocessor scheduling using hybrid particle swarm optimization with dynamically varying inertia. Int J Comput Sci Appl 4(3):95–106
19. Steinbrunn M, Moerkotte G, Kemper A (1997) Heuristic and Ran2 domized optimization for the join ordering problem[J]. VLDB J 6(3):8–17
20. Tong Z, Deng X, Ye F, Basodi S, Xiao X, Pan Y (2020) Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment. Inf Sci 537:116–131 [CrossRef]
21. Xing Z, Jianhua P, Wei Y (2020) A privacy-aware computation offloading method based on Lyapunov optimization. J Electron Inf Technol 42:704–711
22. Xu J, Li X, Ding R, Liu X (2019) Energy efficient multi-resource computation offloading strategy in mobile edge computing
23. Yang L, Zhang H, Li M, Guo J, Ji H (2018) Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans Veh Technol 67:6398–6409 [CrossRef]
24. Yiyi Z, Ye Y, Jinhu L et al (2020) A task offloading algorithm in mobile edge cloud computing. [J] Comput Appl Softw 37(6):135–141
25. Zhang, Y, Yan, L (2022) A fast face recognition based on image gradient compensation for feature description. Multimed Tools Appl. https://doi.org/10.1007/s11042-022-12804-4
26. Zhang H, Guo J, Yang L, Li X, Ji H (2017) Computation offloading considering fronthaul and backhaul in small-cell networks integrated with MEC. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017; pp. 115–120
27. Zhang J, Hu X, Ning Z, Ngai EC-H, Zhou L, Wei J, Cheng J, Hu B (2017) Energy-latency tradeoff for energy-aware offloading in Mobile edge computing networks. IEEE IoT J 5:2633–2645 [CrossRef]
28. Zhou S, Jadoon W (2021) Jointly optimizing offloading decision and bandwidth allocation with energy constraint in Mobile edge computing environment[J]. Computing, (99)
29. Zhu Y (2020) Research on multi-priority task scheduling algorithms for Mobile edge computing [D]. Univ Electron Sci Technol China

Yanhu Zhang born in 1981. Senior engineer. ph. D. His research interest covers algorithm research, edge computing, image recognition (forzyh@163.com).

Lijuan Yan born in 1983. Lecturer. M.S. Her research interest covers algorithm research, image recognition, wireless network(juanjanny@qq.com).