International Journal of Computer Techniques -- Volume 5 Issue 1, Jan – Feb 2018

RESEARCH ARTICLE OPEN ACCESS

Energy Efficiency Multi task Offloading and Resource Allocation in Mobile Edge Computing

LiHuanjie Zang*

* (Department of Computer Science, Jinan University, Guangzhou 510632,China)

****

Abstract:

On edge computing, mobile devices can offload some computing intensive tasks to the cloud so that the time delay and battery losses can be reduced. Different from cloud computing, an edge computing model is under the constraint of radio transmitting bandwidth, power and etc. With regard to most models in presence, each user is assigned to a single mission, transmitting power or local CPU frequency on mobile terminals is deemed to be a constant. Furthermore, energy consumption has a positive correlation with the above two parameters. In a context of multitask, such values could be increased or reduced according to workload to save energy. Additionally, the existing offloading methods are inappropriate if all the compute densities of multiple tasks are high. In this paper, a single-user multi-task with high computing density model is proposed and partial task is offloaded when use the different offload algorithm. Simulated annealing algorithm is the best method to select offloading tasks, which can enhance the offloading ratio and save energy consumption.

Keywords —**edge computing, offloading, simulated annealing.**

----------**********----------

I. INTRODUCTION

With popularization of 5G technology, mobile terminals and the Internet of Things (IoT) enter a new round of rapid progress. However, limited computing resources of mobile devices may restrict user experience. In this case, computing intensive tasks can be offloaded to an edge cloud server by virtue of mobile edge computing (MEC) technology. It means that servers of computing and memory properties are deployed in network access points rather close to users so that they are permitted to offload tasks on the mobile terminal to edge servers to achieve a better service effect. Comparing with traditional cloud, the edge cloud has the capability to reduce transmission delay of tasks substantially. Meanwhile, battery losses incurred by migrating computing intensive tasks to the cloud terminal are considerably lower than those consumed by local processing. . The reason is that energy consumption relies on how many CPU cycles are taken by the corresponding tasks. Nonetheless, performance of a MEC system depends on offloading strategies.

Communication between mobile devices and edge servers needs to occupy wireless channels, which gives rise to extra energy consumption. On this basis, how to make trade-off between time delay and energy consumption is a hot spot of MEC investigations. A majority of current studies start from unlimited cloud resources to think about channel assignment problems. For example, wireless channel assignment is defined as a self-centered crowded game [[1]]. As for [[2]] studied the cooperation game of offloading service providers, where the radio and compute resources were assumed to be managed by different entities separately. Computing based on fine grit has been extensively explored recently. For example, an approach is put forward to make program partitioning parallel or serial to sub-tasks [[3]]. In other words, multiple sub-tasks are assigned with certain bandwidth and computing resources to shorten the ultimate completion time.

A huge number of studies focus on a multi-user single-task type now, that is, a user only executes one task to assign channels for offloading decision, both transmitting power and CPU master frequency keep unchanged. What they thought about is that how to allocate bandwidth resources in energy effective way [[4]-[5]]. By contrast, a program is usually partitioned into multiple tasks to be processed in reality. Not only may a single user have the requirement of executing multiple tasks simultaneously, but it is likely for a mobile device to manage several IoT appliances and perform their tasks. Therefore, transmitting power and CPU frequency should be selected accordingly in line with offloading choices of different tasks in multiple tasks. In a multi-task scenario [[6]], the user is allowed to offload all or partial tasks, which dependent on their transmitting power and noise. In the case that all tasks of the device are processed locally, it no longer carries out radio transmission to the base station, as a result, the relevant transmission power can be 0 or maximum. If a user only needs to offload a small part of tasks, a great transmitting power leads to unnecessary energy consumption. It is suggested that a task can be partitioned into multiple time slots to be executed in [[7]][[8]]. Moreover, some tasks of certain bits are performed locally in each time slot, while some others are offloaded to the server. A user should modify local CPU frequency and transmitting power in conformity with workload to minimize energy consumption [[7]], where some tasks are executed locally and other tasks are executed on the server respectively, but they ignore that the offloading tasks can adjust dynamically. Although the number of executed bits locally or on the server is considered to be modified in [[8]], both transmitting power and CPU frequency cannot be adjusted in accordance with the corresponding workload. Meanwhile, random increase or decrease of bits may damage program integrity due to correlation of programs. Furthermore, a program should be partitioned into several tasks that can be implemented independently. Consequently, how to select some tasks to be offloaded and completed before the deadline to reduce the final energy consumption is an NP-hard problem.

The single-user multi-task model is used in this paper, multiple independent tasks must be executed and completed before the deadline locally or on the server. The existing effective energy utilization model selects tasks of high computing densities to be offloaded, which fails to adapt to scenarios of multiple tasks all with high computing densities. The simulated annealing algorithm is used to an offloading selection decision in this paper. It is applicable to multiple tasks of high computing densities to avoid the achievement of a locally optimal solution in the process of offloading selection. The simulation results show that the model has the potential to improve task offloading ratio and reduce the power consumption of task execution.

Structure of this paper is as follows. Section 1 introduces related works. Section 2 put forward a formulaic definition of the proposed model. In section 3, corresponding problems are transformed into convex optimization problems. An offloading strategy of low time complexity is presented in section 4. For section 5, numerical simulation of different methods is carried out. The paper is concluded in section 6.

II. SYSTEM MODEL

As shown in Fig.1, a multi-task and single mobile terminal model is taken into account in this paper. In this model, a piece of code is divided into multiple mutually independent tasks that are executed locally or offloaded to corresponding edge servers for computing. On the mobile terminal, computation and transmission are implemented concurrently and all tasks must be completed within the time limit. Local and cloud task execution is represented as mathematical definition to compute energy consumption and delay during computing and sending respectively, so as to formulate a power minimization implementation strategy.

A mobile device was assumed to contain N independent tasks, which is denoted as $\mathcal{T} \triangleq \{T_1, \dots T_N\}$. Each task could be expressed in a two-element tuple $< D_i, C_i >$, where D_i (bits) is data size of input data, that consists of environment settings, program code and initial parameters of task execution, C_i stands for CPU cycles of these tasks. Values of D_i and C_i depend on the nature of tasks and are achieved by analyzing concrete task execution situations [[9]-[10]]. For a user, if task i is selected to execute locally, CPU frequency of the

local machine is denoted by F. In this context, local task execution time (s) can be denoted as equation (1).

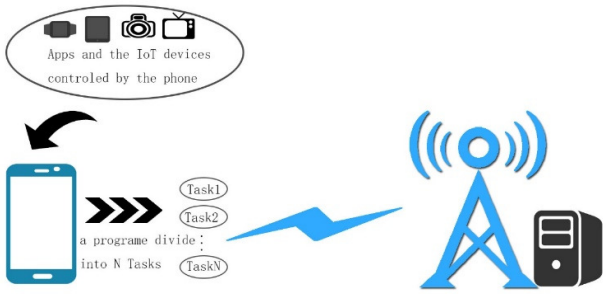

Fig. 1 Edge computing system model

$$t_i^{loc} = C_i/F \tag{1}$$

To calculate power consumption(J) of local task execution, a widely recognized model utilized for energy consumption of CPU cycles has been adopted in this study. To be specific, energy consumption of each CPU cycle is expressed as $\varepsilon = \kappa F^2$ [[1]] [[11]], where, κ is a parameter established according to CPU chip architecture. Therefore, power consumption of local task execution is expressed as equation (2).

$$E_i^{loc} = \kappa F^2 C_i \tag{2}$$

As tasks are uploaded to a MEC server, its total completion time should be composed of three parts, namely (i) t_i^{up} is the duration of transmitting the input data to MEC server via the uplink; (ii) t_i^{exe} denotes the execution time of the tasks on server; and (iii) the time required by the task to return results. Generally, time delay of returning computing results to a user is ignored, because data size obtained by computing is far lower than the input data size. It is consistent with considerations mentioned in [[1]] [[12]]. Here, t_i^{up} is related to transmission rate of the uplink. In line with Shannon equation, the transmission rate(bits/s) can be expressed in the following formula as equation (3).

$$\mathrm{R}(p) = \mathrm{W}\log(1 + \frac{ph^2}{\sigma}) \tag{3}$$

In the equation (3), W stands for upstream bandwidth, h for channel gain jointly determined by the distance from a user to a wireless base station and the path loss factor, p for transmission rate required by a user to upload the input data to a nearby edge cloud server, and σ for a task's background noise including interference of other users' transmission and noise of the natural environment. On this basis, transmission time and computing time of uploading a task can be denoted as the total time delay of this task on the server. If computing resources that a user has access to use is expressed in F^O, the corresponding formula can be written as equation (4).

$$t_i^{up} + t_i^{exe} = \frac{D_i}{\mathrm{R}(p)} + \frac{C_i}{F^O} \tag{4}$$

Considering that data size acquired by cloud processing is far below the input data size, only energy consumption of uplink transmission needs taking into account. In addition, transmitting power of a mobile device has an upper limit dependent on the LTE standard. Hence, energy consumption can be expressed in equations (5) and (6), where P^{Max} signifies the maximum transmitting power.

$$E_i^{off} = pt_i^{up} \tag{5}$$

$$0 < p < \mathrm{P}^{Max} \tag{6}$$

Eventually, calculation model in this study is written into the following equations.

$$\min_{\beta,p,F}\left(\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}\right) \tag{P1}$$

s.t.

$$0 < p < \mathrm{P}^{Max} \tag{6}$$

$$\sum_{i\in\beta_{1,i}} t_i^{loc} < T^d \tag{7}$$

$$\sum_{i\in\beta_{2,i}}\left(t_i^{up} + t_i^{exe}\right) < T^d \tag{8}$$

$$\sum_{i=1}^{N} \alpha_{1,i} * \alpha_{2,i} = 0 \tag{9}$$

$$\sum_{j=1}^{2} \alpha_{j,i} = 1 \tag{10}$$

Task T_i is represented by a binary set of $\alpha_{1,i}, \alpha_{2,i}$, $\alpha_{1,i} = 1, \alpha_{2,i} = 0$ when $i \in \beta_1$; and, $\alpha_{1,i} = 0, \alpha_{2,i} = 1$ in the case of $i \in \beta_2$. In this part, β_1 and β_2 represent a set of tasks executed locally and the other set of tasks implemented on the cloud respectively. The objective function means that transmitting power and chip operating frequency on the mobile terminal should be modified and appropriate tasks are selected on the cloud to minimize the power consumption when relevant time delay has been restricted and the corresponding energy consumption remains below its upper limit. For example, we could use DVFS to make the CPU frequency adjust dynamically. While constraint (6)

indicates that transmitting power should not exceed its upper limit or be a negative, constraints (7) and (8) point out that tasks processed locally or on the cloud must be fully implemented and completed before the deadline. According to constraints (9) and (10), a task must be executed either locally or on a cloud.

P1 is a mixed integer non-linear programming problem(MINLP) to satisfy time delay conditions by modifying transmitting power p and CPU frequency F. In addition, an offloading strategy β should be also formulated to minimize the power consumption, which has been deemed as a NP hard problem. In the next section, complexity of this problem was cut down by means of relaxation and decoupling to find a feasible solution in low time complexity.

III. PROBLEM FORMULATION

To resolve problem P1, it is necessary to reduce its complexity. If the offloading strategy β has been clear to us, constraints (7) and (8) are only related to the first half part and the latter part of P1 separately. After relaxation and decoupling of the problem, the following two sub-problems are achieved by satisfying transmitting power adjustment in a condition of time delay.

$$\min_{p} \sum_{i\in\beta_2} E_i^{off} \qquad \text{(P2)}$$

s.t. (8)

And CPU frequency adjustment subjected to the (7) condition.

$$\min_{F} \sum_{i\in\beta_1} E_i^{loc} \qquad \text{(P3)}$$

s.t. (7)

In terms of problem P2, it can be written as follows in accordance with Equations (3), (4) and (5).

$$\min_{p} \sum_{i\in\beta_2} p\,\frac{D_i}{\mathrm{R}(p)} \qquad \text{(P4)}$$

$$\text{s.t.} \sum_{i\in\beta_2}\left(\frac{D_i}{\mathrm{R}(p)} + \frac{C_i}{F^O}\right) < T^d \qquad (11)$$

Nonetheless, this is still a non-convex problem, because of a non-convex objective function corresponds to a non-linear constraint. To transform it into a convex optimization problem, variable substitution is adopted. In this case, a variable ξ has been introduced to let $\xi = \frac{1}{\mathrm{R}(p)}$ and $Z = \frac{\sigma}{h^2}$, then, the original problem is turned into P5,

$$\min_{\xi} Z * \sum_{i\in\beta_2}\left(2^{\frac{1}{W\xi}} - 1\right)\xi D_i \qquad \text{(P5)}$$

$$\text{s.t.} \sum_{i\in\beta_2}(D_i\xi + C_i/F^O) < T^d \qquad (12)$$

Define that $\Lambda(\xi) \triangleq (2^{\frac{1}{W\xi}} - 1)\xi$, and $\frac{\mathrm{d}^2\Lambda(\xi)}{\mathrm{d}\xi^2} = 2^{\frac{1}{W\xi}}\frac{\ln^2 2}{W^2\xi^3} \geq 0$ corresponding to a condition of $\forall\xi > 0$, here the objective function is a convex function and the constraint condition becomes linear. Therefore, such a convex optimization problem can be solved by a Lagrangian multiplier method in [[13]].Make $\Gamma(\xi,\lambda) = Z * \sum_{i\in\beta_2}\xi(2^{\frac{1}{W\xi}} - 1)D_i + \lambda(\sum_{i\in\beta_2}(D_i\xi + C_i/F^O) - T^d)$, where λ is the Lagrange's multiplier, and $\frac{\partial\Gamma}{\partial\xi} = 0$ and $\frac{\partial\Gamma}{\partial\lambda} = 0$, that is, $2^{\frac{1}{W\xi}} * \left(1 - \frac{\ln 2}{W\xi}\right) - 1 + \lambda = 0$, $\sum_{i\in\beta_2}(D_i\xi + C_i/F^O) - T^d = 0$;That is, $\lambda = 1 - 2^{\frac{1}{W\xi}}\left(1 - \frac{\ln 2}{W\xi}\right) = -\frac{d\Lambda(\xi)}{d\xi}$. Due to $\frac{\mathrm{d}^2\Lambda(\xi)}{\mathrm{d}\xi^2} \geq 0$, $\frac{d\Lambda(\xi)}{d\xi}$ is an increasing function about ξ; together with $\lim_{\xi\to 0}\frac{d\Lambda(\xi)}{d\xi} = -\infty$ and $\lim_{\xi\to+\infty}\frac{d\Lambda(\xi)}{d\xi} = 0$, it can be concluded that $\lambda = -\frac{d\Lambda(\xi)}{d\xi} > 0$ in the case of $\xi \in (0,+\infty)$. Moreover, λ is monotone decreasing in relation with ξ. To obtain the value of λ, this problem is resolved by dichotomy as algorithm1

Algorithm1:Subcarrier-Search For Transmission Power

Input : $D_i \in \beta_2, C_i \in \beta_2, \varepsilon, T^d, W, h^2, \sigma, F^O$

Output: ξ

$\lambda_l = 0$ $\lambda_h = \lambda_{max}$

While

$\lambda_m = (\lambda_l + \lambda_h)/2$

Solve the equation $2^{\frac{1}{W\xi}} * \left(1 - \frac{\ln 2}{W\xi}\right) - 1 = \lambda_m$

If $\left|\sum_{i\in\beta_2}\left(t_i^{up} + t_i^{exe}\right) - T^d\right| < \varepsilon$ **break**

Else If $\sum_{i\in\beta_2}\left(t_i^{up} + t_i^{exe}\right) > T^d$ $\lambda_l = \lambda_m$

Else If $\sum_{i\in\beta_2}\left(t_i^{up} + t_i^{exe}\right) < T^d$ $\lambda_h = \lambda_m$

End If

End While

Likewise, P3 can be also transformed into a convex optimization problem by means of

substitution. The reason why it can be directly solved by the Lagrangian multiplier method is that problem P3 has a convex objective function and a linear constraint.

In detail, it is assumed that $\Lambda(f_{sec},\mu) = \sum_{i\in\beta_1}\kappa F^2\ C_i + \mu(\sum_{i\in\beta_1}\frac{C_i}{F} - T^d)$, to make $\frac{\partial\Lambda}{\partial F} = 2\kappa F C_i - \mu\frac{C_i}{F^2} = 0$ and $\sum_{i\in\beta_1} C_i/F - T^d = 0$, then, $\mu = 2\kappa(F)^3$. Apparently, μ is descending in terms of F in a context of $(0,+\infty)$, where, $F \in (0,+\infty)$. Similarly, value of F can be also gained by dichotomy.

Algorithm2: Subcarrier-Search For CPU Frequency

Input：$D_i \in \beta_1, C_i \in \beta_1, \varepsilon, \mathrm{T^d}, \kappa$
Output: F
$\mu_l = 0\ \ \mu_h = \mu_{max}$
While
$\mu_m = (\mu_l + \mu_h)/2$
Solve the equation $\mu_m = 2\kappa(F)^3$
If $\left|\sum_{\mathrm{i}\in\beta_1}\mathrm{t_i^{loc}} - \mathrm{T^d}\right| < \varepsilon$ **break**
Else If $\sum_{\mathrm{i}\in\beta_1}\mathrm{t_i^{loc}} > \mathrm{T^d}\ \mu_l = \mu_m$
Else If $\sum_{\mathrm{i}\in\beta_1}\mathrm{t_i^{loc}} < \mathrm{T^d}\ \mu_h = \mu_m$
End While

Regarding Algorithm1 and Algorithm2, their time complexity is $O(\log n)$, Equations (2) and (5) are used to work out $\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}$. In the end, the problem lies in the solution to task allocation β.

$$\min_{\beta}\left(\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}\right) \qquad \text{(P6)}$$

s.t. (6),(9),(10)

However, it is still an NP-hard problem, the optimal solution of which can be acquired by listing all possible solutions in a method of exhaustion. Considering that each task can be implemented after being offloaded to a cloud or locally, the number of possibilities is 2^n in total. Time complexity of this decision scheme is $O(2^n \log n)$ that we cannot accept. In the next section, an algorithm of low complexity is proposed to solve this issue.

IV. SOLVING OFFLOAD DECISION-MAKING PROBLEM ON LOW TIME COMPLEXITY

In this part, the simulated annealing algorithm is presented to make offloading decisions so that the problem could be resolved in a condition of polynomial time complexity. For the convenience of description, the following operation has been defined. In the course of this operation, task i is taken out of one set and then put into the other. At the beginning, $\forall i \in \beta_1$ and $\beta_2 = \emptyset$, indicating that all tasks are assumed to be locally executed originally.

Algorithm3:Exchange

Input：β_l, β_r, i
Output：β_l, β_r
If $i \in \beta_l$
$\beta_l = \beta_l \backslash i$
$\beta_r = \beta_r \cup i$
End If

In most research, tasks of high computing density always achieve an offloading priority of a higher level. For example, $E_i^{loc} > E_i^{off}$ has been selected as a condition to pick offloaded tasks to a cloud in multiple papers, for an example, paper [[6]] is inclined to offload tasks of high computing density. Generally, diverse tasks are with different computing densities. If their computing densities are slightly differentiated and also high, energy consumption of the cloud is mainly incurred by data upload, while the local power consumption is computing, migration of tasks to a cloud must save more energy than that executed locally with the condition of urgent deadline, which may lead to the following situations. Because the higher computing densities tasks has the higher offloading priority, the possibility of migration can be higher for tasks which has a small data size but great computational complexity. But such tasks with huge data size as well as high computational complexity will become too late to be transmitted. They have to be completed before the deadline by increasing CPU frequency locally due to a fact that the transmitting power has an upper limit, further resulting is unexpected increasing energy consumption.

On this basis, a greedy offload (GRO) scheme is proposed. It attempts to select a task to offload from all candidate tasks; after the calculation of objective function P6, it is placed back into the original set; then, another task is chosen to offload from other candidate tasks to work out the relevant objective function and then put back again… In this way, a round of offloading is completed for all tasks to find a task of optimal performance and this task is thus

selected for computing migration, followed by the next round up until the offloading decision becomes unable to acquire any other value better than the current function results. The optimal offloading decision of a round is made after all tasks have been calculated and compared, let n the number of rounds, complexity of GRO can be denoted as $O(n^2 \log n)$.

Algorithm4:Greedy Offload(GRO)

Input : $D_i \in \beta_1, C_i \in \beta_1,$
Output: β_1, β_2
Value= $\sum_{i\in\beta_1} E_i^{loc}$
While(true)
  $j = -1$
  For $i \in \beta_1$
    Exchange(β_1, β_2, i)
    tempValue=$\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}$
    If tempValue<Value **And** $p < P_{max}$
      Value= tempValue
      $j =$ i
    End If
    Exchange(β_2, β_1, i)
  End For
  If $j == -1$
    Break
  End If
  Exchange(β_1, β_2, j)
End While

We should pay attention a phenomenon that when the current most power-efficient task has been selected and offloaded, because the selected task occupies great transmitting power such that e other tasks cannot be offloaded; the sum of power consumption of offloading these tasks may be higher than that spared by the present task. For instance, offloading T_1 spares energy 0.1J and consumes the transmitting power 0.11W; by contrast, the energy of offloading T_2 and T_3 is both 0.06J and each of them consumes 0.1W transmitting power. Therefore, when the transmitting power remains below 0.2W, it is not ideal to select T_1 to offload. Such a category of problems can be summarized as 0-1 knapsack problems with corresponding solutions.

In this paper, we assume that offloading and transmitting power consumption of each task is not constant amounts, but both vary along with different task combinations. In this condition, $\left|\sum_{i\in\beta_2}\left(t_i^{up} + t_i^{exe}\right) - T^d\right| < \varepsilon$ is used to compute transmitting power p. Value of $\sum_{i\in\beta_2}\left(t_i^{up} + t_i^{exe}\right)$ is different for different offloading task combinations in a condition of identical λ; hence, value of p is also distinct. In a word, we do not know the transmitting power and energy saving about a tasks until we calculate them according to different β_1, β_2. So we cannot use the solution about 0-1 knapsack problem.

In this section, solution of this problem is improved by the simulated annealing algorithm. Simulated annealing is a term originated in metallurgy. During annealing, the material is heated and cooled down at a particular rate so that it becomes much more likely for atoms to find positions where internal energy is much lower than before. Statistically, optimal solution to simulated annealing converges to a locally optimal solution with probability. In general hill climbing algorithms, they stop when solutions of the near space can be no longer better than the current solution. But, the simulated annealing algorithm accepts a solution poorer than the current one in accordance with a certain probability so that it can possibly step out such a local optimal solution. When the objective function $Function(i+1)$ is better than $Function(i)$, the corresponding solution is accepted; otherwise, the migration should be accepted in conformity with a certain probability that gradually changes along with time.

Computing density is used as a control factor in the proposed algorithm. Under the circumstance of similar data sizes, it will save more energy to offload the task with higher computing density. On this basis, equation (13) is employed to initialize relevant solutions. Let χ_i be the computing density proportion, which is obtained by equation (13), where ρ_{max} *is the* maximum computing density, ρ_{min} is the minimum computing density and ρ_i is computing density of the current task.

$$\chi_i = \frac{\rho_i - \rho_{min}}{\rho_{max} - \rho_{min}} \quad (13)$$

Obviously, $\chi_i \in [0,1]$, χ_i is closer to 1 in the case of a higher computing density. Therefore, only when the sum of χ_i and cooling factor t is below r, r represents the random factor and $r \in [0,1]$, the task can be placed in a local position. In this way, it is much possible that tasks with higher computing densities will be implemented on an edge server.

Subsequently, energy consumption of this scheme is calculated. If the energy consumption is lower than the current value, the scheme can be accepted; otherwise, solutions should be accepted in line with a particular probability by virtue of Equation (14).

$$Pro = \frac{1}{1+e^{-\Delta E_i/T}} \quad (14)$$

Where, T stands for the initial temperature and is a constant, ΔE_i is the difference between the new and the original solutions. In the case, the random number is smaller than Pro, the solution is poorer than the current one, which should be accepted; otherwise, it is rejected.

Eventually, the simulated annealing offload (SAO) algorithm is presented in Algorithm 5.

Dependent on SAO, the probability of executing a task with high computing density locally increases, which makes tasks with large data size and massive calculations be migrated and prevent the low data size and high computational complexity tasks to take the high offload priority. In order to avoid local optimum, a poorer solution than the current one should be accepted in line with a particular probability.

Time complexity of the algorithm depends on an annealing factor fac. The inner loop possesses time complexity $O(n \log n)$ and the number of exterior loops is m , the ultimate time complexity is $O(mn \log n)$. Some simulation results will validate these ideas in section 5.

Algorithm5: Simulated Annealing Offload (SAO)

Input：$D_i \in \beta_1, C_i \in \beta_1, \epsilon, fac, T, \Delta t$
Output: $\beta_{1\_result}, \beta_{2\_result}$
oldValue= $\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}$
While $fac > \epsilon$
  For $i \in \beta_1 \cup \beta_2$
  $Caculate\ \chi_i\ according\ to\ (11)\ and\ random\ r\ between(0,1)$
    If $r \geq \chi_i + fac$
    Exchange(β_2, β_1, i)
    Else If
      Exchange(β_1, β_2, i)
    End If
    newValue= $\sum_{i\in\beta_1} E_i^{loc} + \sum_{i\in\beta_2} E_i^{off}$
    If newValue < oldValue
      oldValue = newValue
      If minValue<old Value
        minValue=oldValue
        $\beta_{2\_result} = \beta_2$
        $\beta_{1\_result} = \beta_1$
      End If
    Else
      $caculate\ Pro\ according\ to\ (12)$
      $and\ Random\ r\ between\ (0,1)$
      If $r < P_i$
        oldValue = newValue
      Else
        $i\ revert\ to\ the$
        $original\ collection$
      End If
    End If
  End For
  $fac = fac - \Delta t$
End While

V. NUMERICAL SIMULATION

In this section, emulation experiment is used to evaluate performance of the proposed algorithm. It is supposed that there is only one mobile device that communicates with an edge server in a community，the mobile device is assigned with bandwidth $W =$ 20MHz, CPU parameter $\kappa = 10^{-28}$[0], channel gain $h = 10^{-3}$, noise $\sigma = 10^{-9}$ and the maximum transmitting power $P_{max} = 0.2$W. In addition, CPU resource F^O on the cloud is 5GHz. During this experiment, data size of a program is assumed to be $[100 \text{ , } 500]$KB roughly, it should be divided into 10 tasks, the data size of each task is denoted as $D_i \in [10 \text{ , } 50]$KB. If all tasks possesses high computing densities, $C_i/D_i \in [1000 \text{ , } 1500]$ cycles/bit, above parameters are all obtained by referring to [[8]]. Then, energy consumption of different methods with diverse time delay is investigated and compared.

First, a high density offload (HDOF) approaches with the priority are compared to GRO in Algorithm3 and SAO in Algorithm 4.. The task offloading should be completed in a descending order of their computing densities until all tasks have been involved. In this course, the transmitting power of a mobile terminal is not allowed to go beyond its maximum value. When they were compared, precision of ε is set to 10^{-3} s when $p\ and\ F$ are

calculated. Particularly, several parameters in the simulated annealing algorithm are set as follows, $fac = 0.5,\ \epsilon = 0.3,\ T = 100$ and $\Delta t = 0.02$ .Corresponding deadline is defined to range from 0.44s to 0.5s.

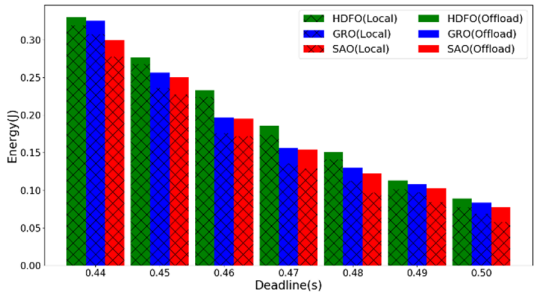

Fig. 2 Energy consumption with different deadline

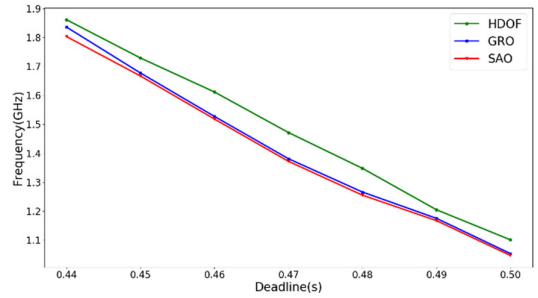

Fig.3 Local CPU frequency with different deadline

Fig. 2 shows energy consumption for HDFO, GRO and SAO approaches at different deadlines, energy consumption goes down along with relaxation of time delay constraints and the average energy consumption of SAO is lower than another two approaches in different deadlines. About 12% and 4% average energy are spared when HDFO and GRO are compared with SAO. Such result signifies that SAO is an ideal scheme. Through vertical comparison, most energy has been consumed by high computing density tasks that cannot be uploaded to the cloud; however, as soon as such tasks have been uploaded successfully, energy consumption decreases evidently.

Due to short transmission time, variations in energy consumed by uploading are small in the case of diverse time delay. Nevertheless, percentage occupied by offloading energy required by all approaches becomes increasingly higher from respectively 3.1%, 5.3% and 7.2% to 10.5%, 10.1% and 17.5% as time delay increases. In conformity with computing, the average uploading energy consumption needed by HDFO, GRO and SAO are 0.026W, 0.043W and 0.055W far below their thresholds, which meets the time delay requirement. If tasks are transmitted by the maximum transmitting power all along, additional energy consumption can be generated. Especially in the case of relaxed deadline, using the maximum transmitting power to translate tasks, the waste of energy consumption will become more and more serious

The comparison between Fig. 2 and Fig. 3 addresses that the CPU frequency depends on performance of energy consumption. The lower CPU master frequency required is, the lower energy consumption will be. Therefore, how to select tasks to be migrated to an edge cloud server is apparently deemed to be especially important. As shown in Fig. 3, SAO can be adopted to effectively select the offloading tasks to lower CPU master frequency, which is 4.8% and 0.9% lower than that consumed by GRO and HDFO separately.

Fig.4 shows offloading ratio of different algorithms at different deadlines, it indicates an interesting phenomenon. When the deadline increases, offloading ratio increases also for all algorithms. The offloading rate of SAO algorithm is lower than HDOF and GRO.

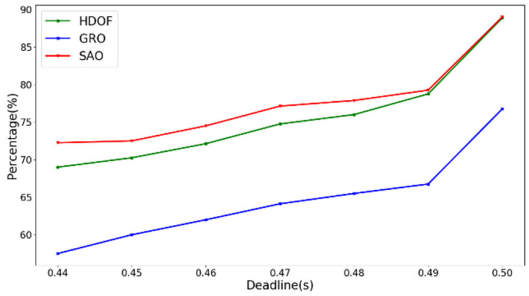

Fig.4 Offload percentage with different deadline

While SAO generates the highest offloading rate over 70%, offloading rate of HDOF with the maximum energy consumption is greater than that of GRO requiring less energy. By contrast, it has been proven in Fig. 2 that its uploading energy consumption is lower than another two approaches. This illustrates that tasks HDOF selects to be transmitted have a small data size, which further verifies the conjecture of this study. The reason why

the phenomenon shown in Fig. 4 takes form is that always selecting prioritized tasks all possessing high computing densities may cause some tasks of large data size and massive calculations to be not offloaded. On the contrary, those of heavy calculation burden, small data size and high computing density can be offloaded to the cloud. Consequently, the latter takes up energy consumption of transmitting so that the former fails to satisfy a condition that the transmitting power must be lower than threshold and these tasks are forced to be implemented locally. In this case, power consumption is incurred due to rather high CPU master frequency.

Uploading energy consumption of GRO is roughly identical to that of SAO but greater than HDOF. It signifies that GRO usually selects tasks of large data size and heavy calculation burden to be offloaded to the cloud sparing a lot of energy. However, due to a large data size of these tasks, transmitting power occupied by them is rather high leading to failure in offloading other tasks and a low offloading rate.

Overcoming defects described above, the simulated annealing approach is able to offload tasks have low data size but big calculations in line with a certain probability as well as the massive calculations and big data size tasks. Meanwhile, it also has the capacity to substitute some local optimal solutions to place an energy conservative task occupying great transmitting power in a local position to obtain the possibility of offloading multiple tasks that may be acquired. Furthermore, the sum of energy spared by such multiple tasks is superior to that produced by offloading just one task. In this context, offloading rate related is at the maximum level. Time complexity of the simulated annealing algorithm is slightly higher than that of another two approaches due to several rounds of iterations.

VI. CONCLUSION

The single-user multiple-task model raised in this paper was adopted to assign tasks to be offloaded by virtue of a simulated annealing algorithm and modify both transmitting power and CPU frequency in line with conditions of these tasks. The aim to reduce the total energy consumption of device by dynamically adjusting these two parameters could be achieved.

As demonstrated by emulation experiment based on numerical simulation, in the case of multiple tasks have high computing densities that offloading them according to computing densities may give rise to failure in offloading those of big data size but heavy calculation burden; besides, keeping selecting tasks that seem to be most energy conservative to be offloaded to the cloud can possibly lead to a low offloading rate and falling into the local optimal solution because such an operation has occupied excessive transmitting power. However, the simulated annealing algorithm proposed in this paper is capable of offloading tasks that consuming much energy executed locally and improving the offloading rate simultaneously to ultimately lower the total energy consumption.

REFERENCES

[1] X. Chen, "Decentralized computation offloading game for mobile cloud computing," IEEE Trans. Parallel Distrib. Syst., vol. 26, no. 4, pp. 974–983, Apr. 2015.

[2] R. Kaewpuang, D. Niyato, P. Wang, and E. Hossain, "A framework for cooperative resource management in mobile cloud computing," IEEE J. Sel. Areas Commun., vol. 31, no. 12, pp. 2685–2700, Dec. 2013.

[3] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Toronto, Canada, Apr. 2014.

[4] M.-H. Chen, B. Liang, and M. Dong, "Joint offloading and resource allocation for computation and communication in mobile cloud with computing access point," in Proc. of IEEE INFOCOM, Atlanta, GA, May 2017.

[5] M. H. Chen, B. Liang, and M. Dong, ''Joint offloading decision and resource allocation for multi-user multi-task mobile cloud,'' in Proc. IEEE Int. Conf. Commun. (ICC), May 2016, pp. 1–6.

[6] K. Liu, J. Peng, H. Li, X. Zhang, and W. Liu, "Multi-device task offloading with time-constraints for energy efficiency in mobile cloud computing," Futur. Gener. Comput. Syst., vol. 64, pp. 1–14, 2016.

[7] Tadapaneni, N. R. (2016). Overview and Opportunities of Edge Computing. Social Science Research Network.

[8] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," IEEE Trans. Wireless Commun., vol. PP, no. 99, 2016.

[9] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," IEEE Trans. Comput., vol. 64, no. 8, pp. 2253–2266, 2015.

[10] A. P. Miettinen and J. K. Nurminen, "Energy efficiency of mobile clients in cloud computing," in Proc. USENIX Conf. Hot Topics Cloud Comput. (HotCloud), June 2010.

[11] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in Proc. IEEE INFOCOM, pp. 2716–2720, 2012.

[12] X. Lyu, H. Tian, P. Zhang, and C. Sengul, "Multi-user joint task offloading and resources optimization in proximate clouds," IEEE Trans. Veh. Technol., vol. PP, no. 99, 2016.

[13] S. Boyd and L. Vandenberghe, *Convex optimization.* Cambridge, UK:Cambridge University Press, 2004.

[14] C. You, K. Huang, and H. Chae, "Energy efficient mobile cloud computing powered by wireless energy transfer," *IEEE Journal on Selected Areas in Communications*.

[15] Liu, Allan and Yu, Ting, Overview of Cloud Storage And Architecture (2018). International Journal of Scientific & Technology Research.

[16] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in Proc. IEEE Global Commun. Conf. (GLOBECOM), Washington, DC, USA, Dec. 2016, pp. 1–6.