TS-SMOSA: A Multi-Objective Optimization Method for Task Scheduling in Mobile Edge Computing

Xuhui Zhao[1, 2], Yan Shi[1], Shanzhi Chen[3]

[1] State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China
[2] Information Engineering College, Henan University of Science and Technology, China
[3] State Key Laboratory of Wireless Mobile Communications, China Academy of Telecommunication Technology, China
haustzxh@163.com, shiyan@bupt.edu.cn, chensz@datanggroup.cn

Abstract

Deploying cloud service at base stations (Cloudlets, considered as Mobile Edge Computing) or sharing resources between mobile devices (Mobile Cloudlets) are expected to be enablers of Edge Computing or Fog computing. Different Mobile Edge Computing systems have been proposed based on hierarchical cloud architecture. However, the dynamics and diversity of mobile cloudlets makes it challenging to offload tasks to optimal nodes (process locally, or offload to a device in Mobile Cloudlets or Cloudlets). In this paper, offloading availability is formulated by considering user mobility similarity and the residual energy of mobile devices. To guarantee offloading availability and minimize energy consumption as well as data size transmitted through the cellular access links, offloading decisions for multi-tasks (task scheduling problem) are formulated to be a multi-objective optimization problem and we develop a centralized algorithm TS-SMOSA such that efficient-energy offloading decisions are made and tasks can be allocated to optimal nodes. Finally, experimental results are promising and show near optimal solutions for all of our studied scenario.

Keywords: Mobile edge computing, Offloading availability, Offloading decisions, Task scheduling, Multi-objective optimization

1 Introduction

To fill the gap between resource-constrained mobile devices and diverse mobile applications, Edge Computing (EC) is proposed as a newly emerging computing paradigm that endow cloud-computing capability at the edge of network (e.g., smartphones, tablets, smart cars, base stations and routers). Deploying cloud service at base stations (Cloudlets, considered as Mobile Edge Computing) or sharing resources between mobile devices (Mobile Cloudlets) are expected to be enablers of EC [1-2]. A similar concept is Fog computing, which is introduced in the context of Internet of Things and big data applications [3]. Therefore, many technologies in EC are also applicable to Fog computing, such as task offloading, resource allocation, mobility management and so on.

In Mobile Edge Computing (MEC), mobile devices can offload tasks to Cloudlet that consists of a number of interconnected physical machines [4]. However, even if considered endowed with computational capacities, Cloudlets cannot offer the same performance as the traditional cloud powerful servers. With the rapidly growing density of mobile devices, the workload in a cloudlet grows bigger following the increment of mobile devices. On the other hand, mobile devices offloading tasks to cloudlet through cellular networks increase the pressure of cellular networks and the growth rate of current cellular system capacity cannot catch up with the demand of mobile applications.

Mobile Cloudlets are assumed to be a promising solution in which a group of nearby mobile devices are connected wirelessly, e.g., using WiFi or Bluetooth. In Mobile Cloudlets, tasks from mobile users can be offloaded to its nearby users that have more powerful computation capabilities. Mobile devices can be providers as well as consumers of service. Potential application scenarios of mobile cloudlets include multimedia sharing at an event, location information acquisition and language translation for a group of tourists [5, 11]. Device-to-Device (D2D) communication is assumed an important technology in the future 5G, which brings Mobile Cloudlets to reality further. Therefore, Mobile Cloudlets are a useful way to leverage resource of mobile device, which provide abundant and efficient resources in an elastic manner.

Different MEC systems have been proposed based on hierarchical cloud architecture [5-8], which aim at cooperatively leverage the hierarchical resource of edge cloud and remote cloud. In this paper, we focus on a 3-tier MEC system, as shown in Figure 1. Specifically, the devices in different tiers have distinct computation and communication capabilities. Tier-1 is mobile cloudlets that are composed of connected

*Corresponding Author: Xuhui Zhao; E-mail: haustzxh@163.com
DOI: 10.3966/160792642019072004006

mobile devices. Tier-2 is Cloudlets, which are deployed at base station or to an edge switch so that multiple base stations can share the computing resources. Tier-3 servers are the existing Cloud Computing infrastructures, such as data centers.

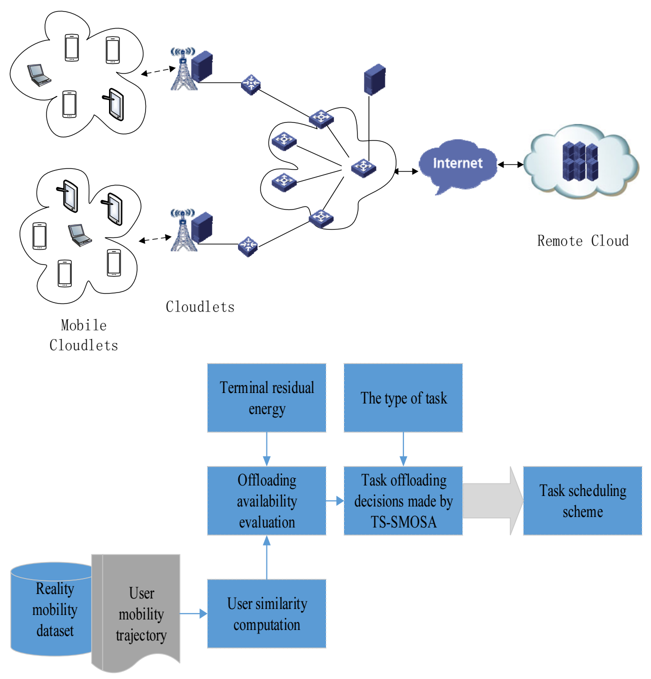

Figure 1. The MEC system model

However, offloading in Mobile Cloudlets is not always successful [9-12]. First, cloud resources in Mobile Cloudlets is dynamic and diverse. As a result, a selfish node may drop an offloading request from other nodes. Moreover, some mobile devices may not be powerful enough to provide energy consumption required for completing an offloaded task, which result in task execution failure. Offloading availability is a complicated problem that is influenced by incentive mechanism, user mobility, the energy of mobile device and resources capacity. Therefore, an optimal provider should be selected in terms of contact possibility (the possibility of accepting offloading requests) and energy efficiency based on the full view of mobile network. Thus, efficient offloading policy considering offloading availability plays an important role to leverage cooperatively hierarchical device resources and cloud resources.

Recent research shows a strong correlation between mobility similarity and social proximity [13-14]. Mobility similarity has a strong indication of friendship [15]. Namely, the higher the similarity of user mobility, the greater the possibility of accepting offloading requests. Thus, in this paper, we consider the effect of user mobility on offloading availability evaluation. Here we define the probability of successfully offloading a task as offloading availability in the system. For a task, we use the term "probability of successfully offloading". Moreover, for the system, we use the term "offloading availability".

To the best of our knowledge, few researches considers offloading availability in MEC system. In this paper, leveraging the advantages of Cloudlets and Mobile Cloudlets, we design offloading policy in hierarchical MEC system. A centralized offloading decision is constructed through synthetically taking the type of task into consideration and user mobility similarity is assumed as a factor in evaluating offloading availability. The detailed model is as shown in Figure 1. To guarantee offloading availability and minimize energy consumption of total mobile devices, as well as data size transmitted through the cellular access links, a multi-objective optimization problem is formulated. Task Scheduling based on Suppapitnarm Multi-Objective Simulated Annealing (TS-SMOSA) is applied to address the formulated problem such that tasks can be offloaded to optimal nodes. Experimental results are promising and show near optimal solutions for all of our studied scenarios.

In particular, our contributions in this paper are:

1. From a new perspective, offloading availability is constructed through synthetically considering user mobility similarity. By computing user mobility similarity based on mobility trajectory data, we use the mobility similarity as a factor in evaluating offloading availability, jointly considers the residual energy of mobile devices.

2. We formally model the problem of task scheduling in MEC as a multi-objective optimization problem, which is to guarantee offloading availability and minimize energy consumption of total mobile devices, as well as data size transmitted through the cellular access links.

3. We develop a centralized algorithm TS-SMOSA such that efficient-energy offloading decisions are made and tasks can be allocated to optimal nodes.

4. We conduct simulation experiments for TS-SMOSA performance evaluation. Experimental results are promising and show near optimal solutions for all of our studied scenarios.

2 Related Work

Mobile Cloud Architecture There have been a lot of research on offloading policy for different kinds of mobile cloud resources architecture [5, 7-8, 16-18]. In [8], two-tier cloud architecture is proposed. Cloudlet is deployed at BS or Wifi hotspot for executing computation-intensive modules and remote cloud for data-intensive modules. [3] present a 3-tier heterogeneous MEC system architecture. A hierarchical MEC is new research trend and it will result in a better distribution of the computing task and a tradeoff between lower energy consumption, execution delay in the system. Specially, with the development of SDN technology, it has been integrated into various networks [16]. [5] proposed a software defined cooperative offloading model that make a decision that mobile devices execute tasks locally, cooperate with other devices or offload tasks to the remote cloud. [6] proposed a network architecture by leveraging the cloudlet concept, the SDN technology, and the cellular network infrastructure. The authors in [11] analyze the availability of Mobile Cloudlets from the theoretical aspect and examine the fundamental properties of Mobile Cloudlets. The authors in [12] propose multi-hop mobile cloudlets and evaluate the performance of computational offloading. All the work above assumed that offloading is always successful in problem formulation. This observation motivates us to consider a more realistic scenario in which offloading decisions for multiple tasks are made at a certain time slot T and offloading availability is taken into consideration.

Task Offloading and Optimization Objectives In most of the existing works, task offloading is formulated into a multi-objective optimization problem which minimize weighted cost, energy and time [9, 19-21], or minimize energy and delay satisfies certain constraints [22-23]. The multi-objective optimization problem is turned into a single-objective problem using a weighted method, which will inevitably miss some feasible solutions. This method usually gets only one optimal solution, which has a great relationship with weight parameters and scalarization method. In practice, it is difficult to determine these parameters. In addition, scalarization method will result in changes of the optimal solution space. Different from the aforementioned work, our goal is to provide a certain amount of well distribution Pareto optima solution and the system can choose one as the task offloading scheme at the current time slot T.

Mobility in MEC Reference [9] developed an offloading algorithm for intermittently connected cloudlet system. User mobility is taken into consideration to evaluate the successful offloading probability. However, the user in [9] is assumed to move straight (the centrifugal direction or centripetal direction) in the cloudlet coverage. Reference [24] proposed an offloading method in which Markov model is used as user mobility model for mobility prediction, but Markov model fail to portray the reality and complex user mobility characteristics. [25] studied the heterogeneous mobile cloud and a centralized algorithm is developed for the offloaded tasks and adopt the most commonly user mobility model, random waypoint to model users' movements. This model fails to describe the human mobility characteristics in real world. The contact and inter-contact durations, also known as the contact and inter-contact time consider the temporal correlation of the user mobility, which are commonly used to model the connectivity between mobile users. This model has been applied in reference [11, 26]. [26] adopts contact duration and inter-contact duration to model user mobility for predicting the probability that two nodes are connected at future time. [27] aims to optimize the mobility and energy charging for mobile cloudlets at the same time. However, the contact and inter-contact time is difficult to obtain in practice, especially in urban scenario. To be more realistic, modern mobile device with GPS has led people to share locations, and these GPS trajectory data mining can reveal commonalities between a pair of users [28].

As shown in [15], although 50% to 70% of human mobility is driven by periodic behavior such as working or going home, 10% to 30% is driven by friendship. Existent analysis shows a strong correlation between social proximity and mobility similarity, namely, mobility between friends are significantly more similar than that between strangers [13]. Therefore, movement trajectories similarity has a strong indication of friendship. It is reasonable to assume that similarity of user mobility indicates the probability that a user accepts offloading request. Based on the observation above, we utilize the similarity of movement trajectories as a factor of offloading availability evaluation. Meanwhile, we also consider the residual energy of mobile devices.

The rest of this paper is organized as follows. Section 3 describes system model and problem formulation. Section 4 devotes to algorithm selection and the proposed TS-SMOSA strategy for multi-task offloading decisions, followed by experiments and results in Section 5. Finally, the paper concluded in Section 6.

3 System Model and Problem Formulation

3.1 Notation

We assume that there are n devices in the network. They have m tasks to be executed, denoted by $T_h = \{T_1, T_2, ..., T_m\}$. We use T_h^u to denote that a task T_h belongs to the user u. A task can be executed locally (mobile device itself), offloaded to other devices (in Mobile Cloudlets), or to the Cloudlet (which is deployed at base station) in accordance with the decision-making in a centralized control manner (e.g. Soft Defined Network Controller). Let $P(\mathrm{T}_h^u)$ denote the offloading decision for a task T_h^u, given by (1), where 0 denotes that task is executed locally, form 1 to n denotes that the task is executed by the certain one worker node in Mobile Cloudlets, and $n+1$ denotes Cloudlet. We define a tuple representation $T_h^u = (Type, \alpha_h, \beta_h)$ to describe the parametric context of each task. $Type$ denotes the type of a task, α_h and β_h denote the input and output data size [5]. Energy consumption for executing tasks includes consumption of computation and transmission consumption. Compared with mobile devices, cloudlets have powerful computing power. Therefore, when a task is offloaded to a cloudlet, we assume that consumption of computation is 0 and energy consumption for executing tasks only includes energy consumption in transmission to cloudlet. Especially, when a task is offloaded to a worker nodes j in mobile cloudlets and the worker node j has already executed the task and cached the results, the energy consumption of j for executing the task with the same *Type* again is close to 0.

$$P(\mathrm{T}_h^u)=\begin{cases}0 & \textit{process locallly}\\ 1,2,...,n & \textit{offload to a device in mobile cloudlet}\\ n+1 & \textit{offload to cloudlet}\end{cases} \quad \textbf{(1)}$$

Based on the given model and scenario, we focus on the following problems: how to schedule multi-user multi-tasks in the two-layer(mobile cloudlets-cloudlets) cloud environment at current time slot T and accomplish all the tasks in offloading service. To design energy-efficient dynamic offloading strategy, a multi-objective optimization problem is formulated from the systematic perspective, which minimize energy consumption of all mobile devices and traffic size in cellular access network, maximize offloading availability. The following describes the three objectives definition under Mobile Cloudlets and Cloudlet execution model.

3.2 Mobile Cloudlets Execution Model

3.2.1 Energy Consumption of Computation

We define the computation energy consumption of device j for executing task T_h^u as (2) [5, 21, 29-30], where X denotes the complexity coefficient of one task. X is the ratio of CPU cycles and the input data size, dependent on the task type. $\mathrm{F}_j()$ is the function of power coefficient of device j's CPU. Here we allow different mobile devices to have different computation capability.

$$E_{compute}^{j}(\mathrm{T}_h^u)=\mathrm{F}_j(\mathrm{X}*\alpha_k) \quad \textbf{(2)}$$

3.2.2 Customer Energy Consumption and Provider Energy Consumption

When the device carried by user u offload task T_h^u to device j, here we define u as the customer and j as the provider. Customer energy consumption and Provider energy consumption can be denoted by (3) (4), where p_t^{uj} (p_r^{uj}) is transmission power of the device carried by user u for data sending(receiving) to (from) j [31], R_T^{uj} is the transmission rate between the link from the device carried by user u and device j at time slot T.

$$\mathrm{E}_{customer}^{uj}(\mathrm{T}_h^u)=p_t^{uj}*\frac{\alpha_k}{R_T^{uj}}+p_r^{uj}*\frac{\beta_k}{R_T^{ju}} \quad \textbf{(3)}$$

$$\mathrm{E}_{\mathrm{Pr}ovider}^{uj}(\mathrm{T}_h^u)=p_t^{ju}*\frac{\alpha_k}{R_T^{uj}}+p_r^{ju}*\frac{\beta_k}{R_T^{ju}}+E_{compute}^{j}(\mathrm{T}_h^u) \quad \textbf{(4)}$$

Therefore, when a task T_h^u is executed in a worker node in Mobile Cloudlets, energy consumption for task completion in the network include customer energy consumption and provider energy consumption.

$$E\ \ (\mathrm{T}_h^u)=\mathrm{E}_{customer}^{uj}(\mathrm{T}_h^u)+\mathrm{E}_{\mathrm{Pr}ovider}^{uj}(\mathrm{T}_h^u) \quad \textbf{(5)}$$

Specially, when $j=u$, it means that device u execute task T_h^u locally. At this point,

$$E\ \ (\mathrm{T}_h^u)=E_{compute}^{u}(\mathrm{T}_h^u) \quad \textbf{(6)}$$

3.2.3 Offloading Availability

Considering the unreliable connection of mobile nodes, offloading availability of mobile cloudlets need to be guaranteed. As shown in [15], a similarity of movement trajectories is a strong indication of friendship. It is reasonable to assume that similarity of user mobility indicates the probability that a user accept offloading request. Based on the observation above, we use the similarity of movement trajectories as a factor in evaluating offloading availability. While user similarity computation is an active area of research [13, 28, 32-33], we simply adopt a lightweight metric called Jaccard coefficient to calculate the similarity of the user's moving trajectories [34-35]. The insights of our work are also useful when other approaches to compute user similarity is adopted.

We use $O=\{1,2,3...,o\}$ to denote the set of location which is visited by all users in a certain period. The user mobility similarity computation depends upon the visited locations in the certain period. Let $\mathrm{O_u}$={a, b, c, d..}, any element belongs to O. For users $u,j\in U$, let η_{uj} denote the user similarity. Then, according to the definition of Jaccard coefficient in [34], we have

$$\eta_{uj}=\frac{|\mathrm{O}_u\cap\mathrm{O}_j|}{|\mathrm{O}_u\cup\mathrm{O}_j|} \quad \textbf{(7)}$$

Then, we define offloading availability $Q(\mathrm{T}_h^u)$ of a task T_h^u as shown in (8), where η_{uj} is user similarity of user u and j, R_j is the residual energy of the provider j.

$$Q(\mathrm{T}_h^u)=\eta_{uj}*\frac{R_j-\mathrm{E}_{\mathrm{Pr}ovider}^{uj}(\mathrm{T}_h^u)}{R_j} \quad \textbf{(8)}$$

Therefore, the average offloading availability of the system can be represented as

$$\frac{1}{m}\sum_{h=1}^{m} Q(\mathrm{T}_h^u) \tag{9}$$

3.3 Cloudlet Execution Model

When a task T_h^u is executed in cloudlet, energy consumption in the network include only transmission consumption between customer and cloudlet, which can be denoted by $\mathrm{E}_{customer}^{uj}(\mathrm{T}_h^u)$. It means that $E\ (\mathrm{T}_h^u) = \mathrm{E}_{customer}^{uj}(\mathrm{T}_h^u)$,where j = n+1. Compared with worker nodes in Mobile Cloudlets, cloudlet deployed at base station have power computation ability and availability. We assume that when a task T_h^u is executed in cloudlet, offloading availability $Q(\mathrm{T}_h^u)$ of a task T_h^u is equal to 1. Specifically, if a task T_h^i is offloaded to another device in Mobile Cloudlets, local execution and cooperative execution between devices do not generate external traffic, which can be expressed by $S(\mathrm{T}_h^u)$ =0. If a task T_h^u is offloaded to cloudlet, the produced data traffic can be expressed as

$$S(\mathrm{T}_h^u) = \alpha_h + \beta_h \tag{10}$$

To this end, with above analytic results on the energy consumption, traffic size and offloading availability, the objective problem can be considered as a multi-objective optimization which involves minimizing energy consumption, minimizing data traffic in access links, maximizing average availability of offloading, which can be denoted as follows:

$$\max \frac{1}{m}\sum_{h=1}^{m} Q(\mathrm{T}_h^u) \tag{11}$$

$$\min \sum_{h=1}^{m} E(\mathrm{T}_h^u) \tag{12}$$

$$\min \sum_{h=1}^{m} S(\mathrm{T}_h^u) \tag{13}$$

4 Algorithm Selection and Design

4.1 Algorithm Selection

If the position sequence of all task being executed is assumed as a vector, while the total energy consumption, traffic usage and offloading availability are analyzed as the three objective functions, the task scheduling problem in this paper can be transformed into a multi-objective optimization problem. Comparing with the single object problems and the weighted multi-objective problem that has one objective function, the multi-objective solution may be not good in each objective, but our goal is to provide a certain amount of well distribution Pareto optima solution and the system can choose one as the task offloading scheme at the current time slot T.

In this paper, based on three objective functions, we can use multi-objective simulated annealing algorithm iteratively to obtain the Pareto solution set of variables. SMOSA is a multi-objective algorithm proposed by Suppapitnarm and based on Simulated Appealing (SA), which is simple to formulate and it can handle mixed discrete and continuous problem with ease [36]. It is also efficient and has low memory requirement. SMOSA takes less CPU time when it is used to solve optimization problems, because it finds the optimal solution using point-by-point iteration [37-39].

Thus, this paper proposes a centralized Task Scheduling algorithm based on Suppapitnarm Multi-Objective Simulated Appealing (TS-SMOSA) to solve the optimization problem proposed above, which can achieve an approximately optimal solution set.

4.2 Algorithm Design

Different from traditional SA method, return-to-base is adopted in TS-SMOSA which enables the search to restart from a randomly selected solution from the Pareto set. A new acceptance probability formulation based on an annealing schedule with multiple temperatures (one for each objective) is also proposed. The key probability step is given as (14). The flowchart of SMOSA is given in Figure 2.

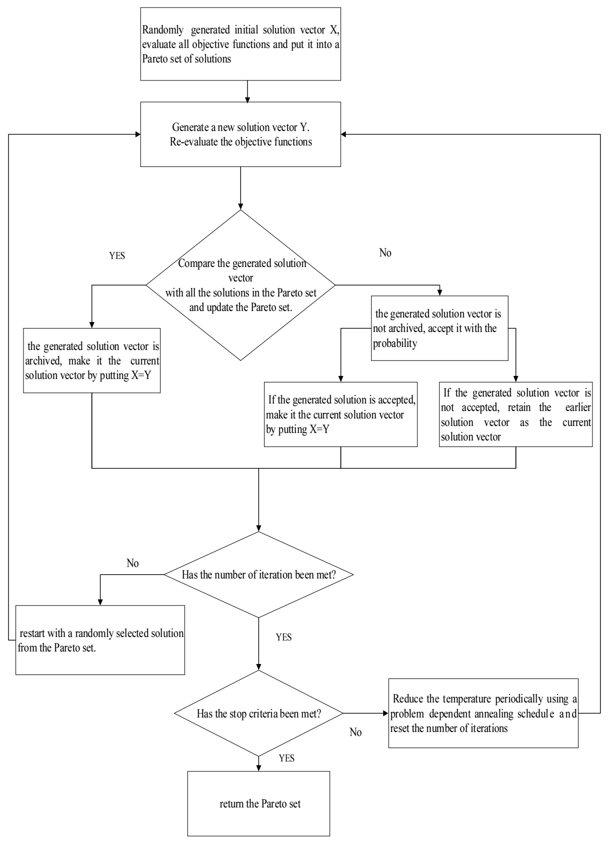
Randomly generated initial solution vector X, evaluate all objective functions and put it into a Pareto set of solutions
Generate a new solution vector Y. Re-evaluate the objective functions
YES
Compare the generated solution vector with all the solutions in the Pareto set and update the Pareto set.
No
the generated solution vector is archived, make it the current solution vector by putting X=Y
the generated solution vector is not archived, accept it with the probability
If the generated solution is accepted, make it the current solution vector by putting X=Y
If the generated solution vector is not accepted, retain the earlier solution vector as the current solution vector
No
Has the number of iteration been met?
restart with a randomly selected solution from the Pareto set.
YES
Has the stop criteria been met?
No
Reduce the temperature periodically using a problem dependent annealing schedule and reset the number of iterations
YES
return the Pareto set

Figure 2. The flowchart of SMOSA

$$P = \min(1, \prod_{i=1}^{N} \exp\left\{\frac{-\Delta s_i}{T_i}\right\}) \qquad \textbf{(14)}$$

The detailed TS-SMOSA algorithm is described as follows. And then we will analyze the effects of various parameters on the experimental performance.

(1) Initialization (Algorithm 1): we randomly allocate a position (a worker node) for each task to be executed. According to (11) (12) (13), all objective functions are calculated and be put into a Pareto set of solutions. The values of the objective functions are respectively represented as f_1, f_2, f_3.

Algorithm 1. Init()

Input: the number of tasks TaskSize, the number of device DeviceSize.
Output: the initial position set for tasks processing.

```
1.  Variable: currentSol: the current solution;
2.  tempSol: a temporal solution;
3.  solResult: Pareto set of solutions;
4.  $f_1$: the value of energy consumption;
5.  $f_2$: traffic size;
6.  $f_3$: the value of availability;
7.  ETAValue: The value of the objective functions
    corresponding to Pareto set.
8.  End Variable
9.  global currentSol //Declare a global variable to
    save the current solution
10. tempSol = [] //define a variable to save a temporal
    solution
11. for $i$ from $0$ to TaskSize do
12. p = random.randint (0, DeviceSize+1) //randomly
    assign a position for each task
13. end for
14. add p into tempSol
15. currentSol = tempSol
16. add tempSol into result set solResults
17. compute values of the objective functions and
    denoted as $f_1, f_2, f_3$.
18. add [$f_1, f_2, f_3$] into ETAValue
```

(2) Generate a new solution (Algorithm 2): randomly select two tasks in current solution and exchange their positions for execution, then take the new position set as a new solution.

Algorithm 2. getNewSolution(ssol)

Input: a position set for tasks processing.
Output: a new position set for tasks processing.

```
1.  Variable: ssol: a position set for tasks processing;
2.  sol: a temporal variable
3.  numList: two random integers
4.  $f1$: the value of energy consumption
5.  $f2$: traffic size
6.  $f3$: the value of availability
7.  ETAValue: The value of the objective function
    corresponding to Pareto set
8.  End Variable
9.  sol = [];
10. for $i$ in range (TaskSize) do
11. sol[i] = ssol[i]
12. end for
13. ssol = sol
14. numList = random.sample (range (0, TaskSize), 2);
    // randomly select two tasks
15. exchange their positions for execution
16. return ssol
```

(3) Judge whether to accept a new solution with multiple temperatures (Algorithm 3): the values of the objective functions corresponding to the new solutions are respectively f'_1, f'_2, f'_3. If $f'_1 < f_1$ and $f'_2 < f_2$ and $f'_3 < f_3$, accept it as the current solution vector. Otherwise, accept it with the probability (14).

Algorithm 3. accept (newSol, t_energy, t_traffic, t_availability)

Input: a position set for tasks processing, t_energy, t_traffic, t_availability.
Output: True or False.

```
1.  Variable: newSol: a position set for tasks
    processing;
2.  t_energy: the temperature of energy consumption
    function
3.  t_traffic: the temperature of traffic usage function
4.  t_availability: the temperature of availability
    function
5.  $f_1$: energy consumption value of current solution
6.  $f_2$: traffic usage value of current solution
7.  $f3$: availability value of current solution
8.  $f_1'$: energy consumption value of new solution
9.  $f_2'$: traffic usage value of new solution
10. $f_3'$: availability value of new solution
11. availThreshod: the minimal availability value
12. End Variable
13. global currentSol;
14. compute values of the objective functions
    according to currentSol and denoted as $f1; f2; f3$;
15. compute values of the objective functions
    according to newSol and denoted as $f_1', f_2', f_3'$;
16. if ($f_1' < f1$ and $f_2' < f2$ and $f_3' < f3$) then
17. return True;
18. else
19. if (($f_1' > f_1$ and $f_2' < f_2$) or ($f_1' < f1$ and $f_2' > f_2$) or
    ($f_3' >$ availThreshod and $f_3 >$ availThreshod)) then
20. energy = abs ($f_1 - f_1'$); //energy difference between
    current and new solution
21. traffic = abs ($f_2' - f_2$); // traffic size difference
22. avail = abs ($f_3 - f_3'$); //availability difference
23. dif=0-(energy*1.0/ t_energy +traffic*1.0/ t_traffic
    +avail*1.0/ t_availability)
24. edif = math.exp(dif);
25. prob = min(1,edif); //compute the accept probability
26. if (random.random() < prob) then
```

```
27. return True;
28. end if
29. else
30. return False;
31. end if
32. return False;
34. end if
```

(4) TS-SMOSA algorithm (Algorithm 4) is a stochastic optimization algorithm based on iteration, it starts from a higher initial temperature; generate a new solution through randomly selecting two task in current solution and exchanging their position on the basis of old solution. With decreasing temperature during repeated sampling, the optimum solutions are obtained. The TS-SMOSA process is described in the following steps: (1) start with a randomly generated initial solution vector, the objective function value is calculated, and assumed as the current optimal solution; (2) according to the new vector generated from current optimal solution, calculate the new objective function values, and determine whether to accept; 3) under different temperatures corresponding to each objective function, perform a certain length of iteration. Then based on annealing schedules, decrease the temperature respectively until the temperature is less than given value thresholds. The detailed process is described as Algorithm 4.

Algorithm 4. TS-SMOSA()

```
Input:
Output: Pareto solution set and the corresponding
        objective function values.
1.  Variable: t_energy: the temperature for energy
    consumption function
2.  t_traffic: the temperature for traffic usage function
3.  t_availability: the temperature for availability
    function
4.  t_min_energy: the minimal value for energy
    consumption
5.  t_min_traffic: the minimal value for traffic usage
6.  t_min_avail: the minimal value for availability
7.  num: the length of iteration in the same
    temperature
8.  r: a random integer
9.  x: annealing rate
10. availThreshod: the minimal availability value
11. ETAValue: the value of the objective function
    corresponding to Pareto set
12. End Variable
13. set initial temperature of energy consumption
    function, traffic usage function, and availability
    function;
14. set annealing rate and minimal temperature of the
    three functions;
15. set the length of iteration in a same temperature
16. init();
17. while (t_energy > t_min_energy and t_traffic >
    t_min_traffic and t_availability > t_min_avail) do
18. for (i in range(num)) do
19. newSol = getNewSolution(currentSol);
20. end for
21. if (accept (newSol, t_energy, t_traffic,t_avail)) then
22. currentSol = newSol;
23. add currentSol into Pareto set solResult;
24. add objective function values corresponding to
    currentSol into ETAValue
25. else
26. r = random.randint(0,len(solResults)-1);
    //randomly select a solution from Pareto set as
    current solution
27. currentSol = solResults[r];
28. end if
29. t_energy = t_energy * x //decrease the temperature
30. t_traffic=t_traffic*x
31. t_avail=t_avail*x
32. end while
33. Return ETAValue
```

5 Experiments

5.1 User Similarity Computation

Experiments using Geolife dataset [40] are conducted to evaluation user trajectory similarity. Geolife is a GPS trajectory dataset containing 182 users collected by Microsoft Research Asia for their Geolife project. A GPS trajectory in this dataset is represented by a sequence of time-stamped points, each of which contains the information of latitude, longitude and altitude. All data processing is carried out in the MySQL database. We observe that sixty percent of the users have fewer than 50 thousands trajectory points. We consider these users as inactive user. For experiments, we selected 31 users who have trajectory points between 50 thousands and 150 thousands. And we use the standard approach to extract stay points for every user. The standard approach towards extracting stay points is to first identify the stay regions and then select the stay points from it. We assume that a stay region is the geographical cluster where a user stays for a period of 30 minutes bounded by a distance of 200 m. For all the users, the stay points are stored in a table. In order to extract the common stay points of different users, we assume that the points within 200m as the same stay point marked with the same number. So we can get the final stay points information. For example, the stay points of users 002-004 are shown in Table 1.

Table 1. the stay points of users 002-004

User _id	Location _id	latitude	longitude	Record_time
002	1	39.899629	116.383413	2008/ 10/31 3:47
002	3	39.898716	116.350065	2008/ 11/1 1:01
002	4	39.90021	116.245314	2008/ 11/1 3:19
003	2	39.907938	116.36848	2008/ 10/31 9:31
003	4	39.910378	116.367675	2008/ 10/31 9:39
004	3	40.000586	116.328166	2008/ 11/3 16:56
004	3	40.010634	116.320839	2008/ 11/5 0:10
004	4	40.00517	116.317553	2008/ 11/5 4:20

According to equation (7), we can get the similarity matrix of user 002,003,004 as shown below. In the same way, the similarity of any two users can be calculated.

$$\begin{bmatrix} 1 & 0.25 & 0.6 \\ 0.25 & 1 & 0.3 \\ 0.6 & 0.3 & 1 \end{bmatrix}$$

5.2 Performance Evaluation of TS-SMOSA

In this section, we consider the urban scenario that a Mobile Cloudlets network with 10 mobile devices. These

device-holders encounter and can communicate with each other. In current time slot T, there are 10 tasks that belong to these devices to be executed. For the cloudlet deployed in base station, uplink and downlink throughput is set to be 2 Mbps and 9 Mbps respectively. We refer to the standard of the transmission rate on link (i, j) and the transmission power measured in [29]. The residual energy of devices a uniform distribution over (11000, 30000) Joule.

We evaluate the performance of the proposed algorithm TS-SMOSA. Randomly Offload (RO, all tasks randomly select a worker node to offload), All Local execution (AL, all tasks are executed on the mobile devices locally) are taken as the performance reference for comparison on the same topology with the same parameter setting. All the algorithms are implemented in Python 2.7.

5.2.1 Impact of Complexity Coefficient

XTo validate the efficiency of our algorithm for different types of tasks, we present the energy consumption, traffic usage, availability in our algorithm for different values X in Figure 3, Figure 4, Figure 5, Figure 6 and Figure 7, where ID represent the identifier number of result data. We generate the input/output data size using two normal distribution N (μ_1, σ_1^2) and N (μ_2, σ_2^2) [41], where the mean μ_1= 10MB, μ_2 = 100MB, and the standard deviation σ_1 = 3 and σ_2=50. We set X to 100, 2000, 3700 Cycle/Byte respectively, which represent most of the typical cases of mobile applications. To evaluate the effectiveness of TS-SMOSA, we randomly select 50 solutions from Pareto solutions obtained from TS-SMOSA and compared with running RO 50 times in the following evaluation. As TS-SMOSA is probability-based algorithm, we run 10 or 20 groups of tests to draw concrete results. The cooling parameter is set to 0.99 and the initial temperature and terminal temperature corresponding to the energy consumption, traffic usage, availability is set to (10000,10), (1000,1), (100,0.01).

The results are shown in Figure 3, Figure 4 and Figure 5, Figure 6 and Figure 7. To the availability, TS-SMOSA always outperforms RO in all our experiments. When X=100, TS-SMOSA and RO consumes more energy than local execution for all tasks. This indicates that the energy consumption generated by communication is greater than that of computation. In this case, local execution for all tasks are more efficient than offloading to a worker node in mobile cloudlets or cloudlets. Offloading in mobile cloudlets or cloudlet is more suitable for computation-intensive tasks. Since task scheduling solution generated by TS-SMOSA and AL have no data transmission through cellular network, the traffic size in both solutions is 0 and availability of the system is set to 1. As our algorithm TS-SMOSA takes availability into consideration, all the solutions in Pareto set generated from TS-SMOSA always outperform RO. When X=1000 and X=2000, TS-SMOSA and RO begin to consumes less energy than AL. To the energy consumption and traffic size, TS-SMOSA and RO has similar performance. When X= 3000 or X=3700, TS-SMOSA and RO consumes less than 30.2% energy than AL, which means that TS-SMOSA can achieve the optimal solution for computation intensive tasks. This is because that if the same type of task has been executed by other devices; our solution has efficiently saved the energy consumption generated by computation.

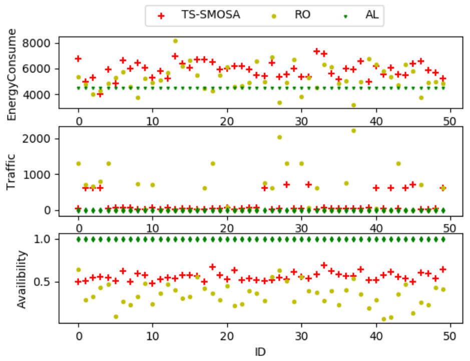

Figure 3. Comparisons of TS-SMOSA, RO and AL (X=100)

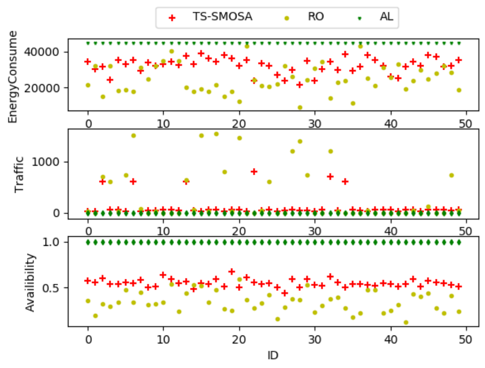

Figure 4. Comparisons of TS-SMOSA, RO and AL (X=1000)

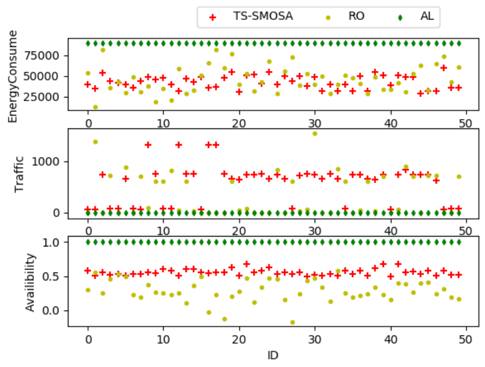

Figure 5. Comparisons of TS-SMOSA, RO and AL (X=2000)

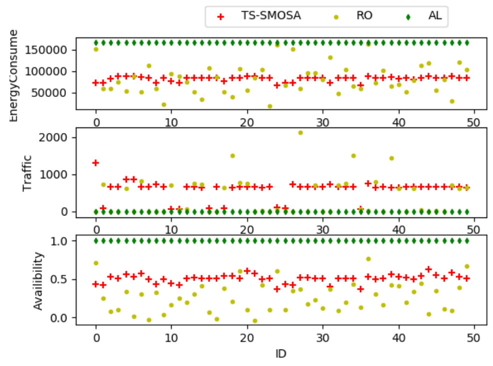

Figure 6. Comparisons of TS-SMOSA, RO and AL (X=3000)

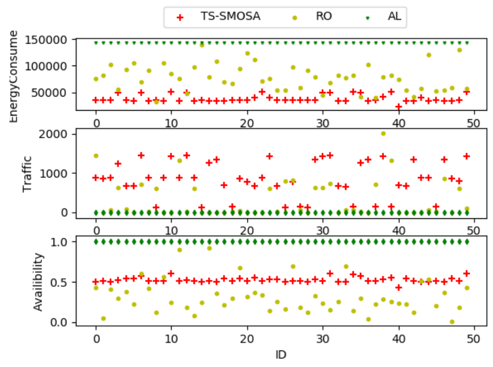

Figure 7. Comparisons of TS-SMOSA, RO and AL (X=3700)

5.2.2 The Impact of TS-SMOSA Cooling Parameter

Since SMOSA is one of the algorithms which are Simulation Annealing-based multi-objective algorithms to find a Pareto set of solutions in a short time, parameters of affecting SMOSA performance include initial value of temperature, cooling schedule, number of iterations, stopping criteria. Based on settings mentioned before, initial value of temperature and stopping criteria eventually affect number of iterations. Therefore, in this article, we discuss the impact of the cooling parameter SPEED and number of iterations on algorithm performance.

We use the same setting on input/output data size as that in the previous section. The initial temperature and terminal temperature corresponding to the energy consumption, traffic usage, availability is also set to (10000, 10), (1000, 1), (100, 0.01), respectively. We set complexity coefficient to 2000. Tests are executed ten times and results are averaged.

The experiment results are shown in Figure 8. Since a larger value of SPEED help the TS-SMOSA to converge, it further reduces the total energy consumption and improved availability of offloading. On the other hand, when the value of SPEED < 0.65, the iterations may not be enough for the TS-SMOSA process to converge, incurring higher energy consumption and traffic size. At the same time, availability is not guaranteed. Besides, due to the speed-up of the cooling process, it is unlikely to accept a temporary solution, Figure 8 shows that when SPEED > 0.8, the iteration of traffic usage begin to converge. However, a higher value of SPEED leads to a larger number of iterations as shown in Figure 9. Because when the value of SPEED increases, the temperature needs more steps to be cooled down to the terminating temperature. As a result, the size of Pareto sets become large. In the previous experiment, we randomly selected 50 solutions to compare with RO.

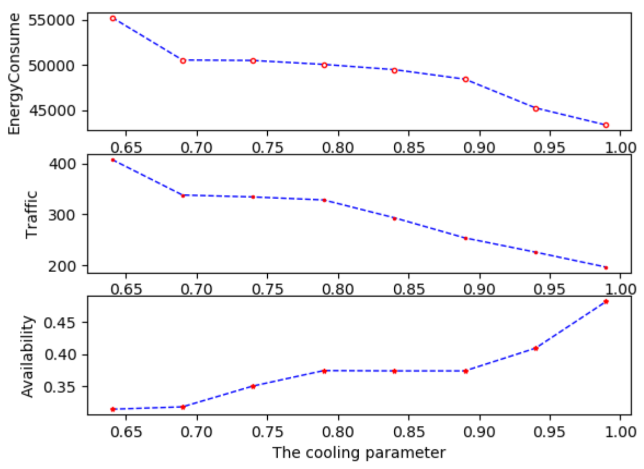

Figure 8. Impacts of cooling parameter SPEED on the performance of TS-SMOSA

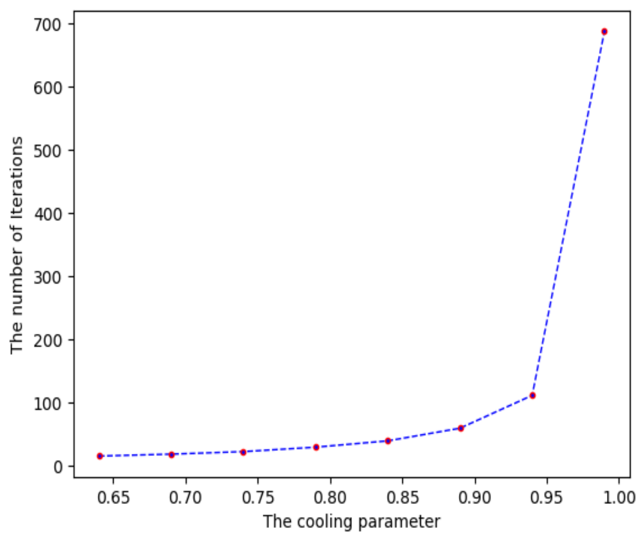

Figure 9. Impacts of cooling parameter SPEED on the number of iterations

The results from Figure 8 and Figure 9 show that there exists a tradeoff between the performance and optimality of task offloading solutions. Using a larger value of cooling parameter helps the TS-SMOSA process converge, reduces the energy consumption and improves offloading availability in the system, but also incurs a larger number of iterations and increases more the computational overhead. In practice, our design offers the system administrator the full flexibility to adjust such parameter as well as optimal solution size at run-time.

6 Conclusion

We define offloading decisions for multi-tasks as an optimization problem with the multi-objectives, i.e. energy consumption of mobile device, traffic size and average offloading availability in the system that consider user mobility similarity and energy efficiency of mobile devices as metrics to evaluate available surrounding users for offloading. These objectives proposed in this paper reflect the key factors when the system administrator making offloading policy for tasks. The TS-SMOSA algorithm is proposed to solve task scheduling problem,

and obtains the approximate optimal solutions. Simulation results show that the multiple solutions for all of our studied cases almost can archive part of the Pareto optimum and is more effective than randomly offloading and local execution. In the future work, we will try to design task migration solution between cloudlet and remote cloud in MEC system.

Acknowledgments

This work was supported by the National Science and Technology Major Projects of China under Grant 2017ZX03001014, the National Natural Science Foundation of China under Grant 61731004, the National Science Fund for Distinguished Young Scholars under Grant 61425012 and Key project plan of Henan Province under Grant 162102210047.

References

[1] Y. Y. Mao, C. S. You, J. Zhang, K. B. Huang, K. B. Letaief, A Survey on Mobile Edge Computing: The Communication Perspective, *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 4, pp. 2322-2358, August, 2017.

[2] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. W. Yang, W. B. Wang, A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications, *IEEE Access*, Vol. 5, No. 99, pp. 6757-6779, March, 2017.

[3] S. Sarkar, S. Misra, Theoretical Modelling of Fog Computing: A Green Computing Paradigm to Support Iot Applications, *IET Networks*, Vol. 5, No. 3, pp. 23-29, March, 2016.

[4] P. Mach, Z. Becvar, Mobile Edge Computing: A Survey on Architecture and Computation Offloading. *IEEE Communications Surveys & Tutorials*, Vol. 19, No. 3, pp. 1628-1656, March, 2017.

[5] Y. Cui, J. Song, K. Ren, M. M. Li, Z. P. Li, Q. M. Ren, Y. J. Zhang, Software Defined Cooperative Offloading for Mobile Cloudlets, *IEEE/ACM Transactions on Networking*, Vol. 25, No. 3, pp. 1746-1760, February, 2017.

[6] X. Sun, N. Ansari, Latency Aware Workload Offloading in the Cloudlet Network, *IEEE Communications Letters*, Vol. 21, No. 7, pp. 1481-1484, April, 2017.

[7] N. Cheng, W. C. Xu, W. S. Shi, Y. Zhou, N. Lu, H. B. Zhou, X. M. Shen, Air-Ground Integrated Mobile Edge Networks: Architecture, Challenges and Opportunities, *IEEE Communication Magazine*, Vol. PP, No.99, pp. 1-7, 2018.

[8] W. P. Ouyang, Y. L. Teng, M. Song, W. X. Zhao, Queue-aware Energy Minimisation through Sparse Beamforming in C-RAN, *IET Communications*, Vol. 12, No. 5, pp. 559-565, March, 2018.

[9] Y. Zhang, D. Niyato, P. Wang, Offloading in Mobile Cloudlet Systems with Intermittent Connectivity, *IEEE Transactions on Mobile Computing*, Vol. 14, No.12, pp. 2516-2529, February, 2015.

[10] Q. Qi, J. X. Liao, J. Y. Wang, Q. L, Y. F. Cao, Dynamic Resource Orchestration for Multi-Task Application in Heterogeneous Mobile Cloud Computing, *IEEE Conference on Computer Communications Workshops*, San Francisco, California, 2016, pp. 221-226.

[11] Y. J. Li, W. Y. Wang, Can Mobile Cloudlets Support Mobile Applications? *International Conference on Computer Communications*, Toronto, Ontario, 2014, pp. 1060-1068.

[12] C. Funai, C. Tapparello, W. Heinzelman, Mobile to Mobile Computational Offloading in Multi-Hop Cooperative Networks, *IEEE Global Communications Conference*, Washington, District of Columbia, 2017, pp. 1-7.

[13] C. Fan, Y. D. Liu, J. M. Huang, Z. H. Rong, T. Zhou, Correlation between Social Proximity and Mobility Similarity, *Scientific Reports*, 2016.

[14] D. S. Wang, D. Pedreschi, C. M. Song, F. Giannotti, A. L. Barabasi, Human Mobility, Social Ties, and Link Prediction, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 2011, pp. 1100-1108.

[15] E. Cho, S. A. Myers, J. Leskovec, Friendship and Mobility: User Movement in Location-based Social Networks, *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Diego, CA, 2011, pp. 1082-1090.

[16] W. Quan, Y. N. Liu, H. R. K. Zhang, S. Yu, Enhancing Crowd Collaborations for Software Defined Vehicular Networks, *IEEE Communication Magazine*, Vol. 55, No. 8, pp. 80-86, August, 2017.

[17] K. Peng, R. H. Lin, B. B. Huang, H. Zou, F. C. Yang, Link Importance Evaluation of Data Center Network Based on Maximum Flow, *Journal of Internet Technology*, Vol.18, No. 1, pp. 23-31, January, 2017.

[18] C. Li, L. Y. Li, Efficient Mobile Cloud Service Allocation for Mobile Commerce, *International Journal of Ad Hoc & Ubiquitous Computing*, Vol. 26, No. 2, pp. 71-80, January, 2017.

[19] H. S. Mounine, H. Ouassila, B. Zizette, Load Balancing, Cost and Response Time Minimization Issues in Agent Based Multi Cloud Service Composition, *International Journal of Internet Protocol Technology*, Vol. 10, No. 1, pp. 10-15, January, 2017.

[20] J. Y. Wang, J. Peng, Y. H. Wei, D. D. Liu, J. L. Fu, Adaptive Application Offloading Decision and Transmission Scheduling for Mobile Cloud Computing, *China Communications*, Vol. 14, No. 3, pp. 169-181, March, 2017.

[21] M. H. Chen, B. Liang, M. Dong, Joint Offloading Decision and Resource Allocation for Multi-User Multi-Task Mobile Cloud, *IEEE International Conference on Communication*, Kuala Lumpur, Malaysia, 2016.

[22] H. Shahzad, T. H. Szymanski, A Dynamic Programming Offloading Algorithm Using Biased Randomization, *International Conference on Cloud Computing*, San Francisco, CA, 2017, pp. 960-965.

[23] K. Zhang, Y. M. Mao, S. P. Leng, Q. X. Zhao, L. J. Li, X. Peng, P. Lan, S. Maharjan, Y. Zhang, Energy-Efficient Offloading for Mobile Edge Computing in 5G Heterogeneous Networks, *IEEE Access*, Vol. 4, No. 99, pp. 5896-5907, August, 2016.

[24] K. Lee, I. Shin, User Mobility-aware Decision Making For Mobile Computation Offloading, *2013 IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications*, Taipei, Taiwan, 2013, pp. 116-119.

[25] Z. Y. Wu, L. Gui, J. C. Chen, H. B. Zhou, F. Hou, Mobile Cloudlet Assisted Computation Offloading in Heterogeneous Mobile Cloud, *International Conference on Wireless Communications Signal Processing*, Yangzhou, China, 2016, pp. 1-6.

[26] Z. H. Wang, H. Shah-Mansouri, V. Wong, How to Download More Data from Neighbors? A Metric for D2D Data Offloading Opportunity, *IEEE Transactions on Mobile Computing*, Vol. 16, No. 6, pp. 1658-1675, August, 2016.

[27] Y. Sui, X. M. Wang, M. Pent, N. An, Optimizing Mobility and Energy Charging for Mobile Cloudlet, *IEEE International Conference on Communications*, Paris, France, 2017, pp. 1-6.

[28] P. Mazumdar, B. K. Patra, R. Lock, S. B. Korra, An Approach to Compute User Similarity for GPS Applications, *Knowledge-based Systems*, Vol. 113, No. 12, pp. 125-142, December, 2016.

[29] A. P. Miettinen, J. K. Nurminen, Energy Efficiency of Mobile Clients in Cloud Computing, *Usenix Conference on Hot Topics in Cloud Computing*, Boston, Massachusetts, 2010, pp. 4-5.

[30] Y. Wen, W. Zhang, H. Luo, Energy-optimal Mobile Application Execution: Taming Resource-Poor Mobile Devices with Cloud Clones, *IEEE Conference on Computer Communications*, Orlando, Florida, 2012, pp. 2716-2720.

[31] K. Kumar, Y. H. Lu, Cloud Computing for Mobile Users: Can Offloading Computation Save Energy, *Computer*, Vol. 43, No. 4, pp. 51-56, April, 2010.

[32] Y. Wan, C. Zhou, T. Pei, Semantic-Geographic Trajectory Pattern Mining Based on a New Similarity Measurement, *International Journal of Geo-Information*, Vol. 6, No. 7, pp. 212, July, 2017.

[33] J. P. Bagrow, Y. R. Lin, Spatiotemporal features Of Human Mobility, http://arxiv.org/abs/1202.0224, 2012.

[34] C. D. Manning, P. Raghavan, H. Schutze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[35] D. Lin, An Information-Theoretic Definition of Similarity, *The Fifteenth International Conference on Machine Learning*, San Francisco, CA, 1998, pp. 296-304.

[36] A. Suppapitnarm, K. A. Seffen, G. T. Parks, J. Kalagnanam, H. Chang, Simulated Annealing: An Alternative Approach to True Multi-objective Optimization, *Molecular Pharmacology*, Vol. 87, No. 2, pp. 314-322, May, 1999.

[37] M. C. Zhang, M. Y. Yang, Q. T. Wu, J. L. Zhu, Smart Perception and Autonomic Optimization: A Novel Bio-inspired Hybrid Routing Protocol for Manets, *Future Generation Computer Systems*, Vol. 81, pp. 505-513, April, 2018.

[38] M. C. Zhang, C. Q. Xu, J. F. Guan, R. J. Zheng, Q. T. Wu, H. K. Zhang, A Novel Physarum-inspired Routing Protocol for Wireless Sensor Networks, *International Journal of Distributed Sensor Networks*, Article number 483581, 2013.

[39] B. Suman, P. Kumar, A Survey of Simulated Annealing as a Tool for Single and Multiobjective Optimization, *Journal of the Operational Research Society*, Vol. 57, No. 10, pp. 1143-1160, October, 2006.

[40] Y. Zheng, L. Z. Zhang, X. Xie, W. Y. Ma, Mining Interesting Locations and Travel Sequences from GPS Trajectories, *Proceedings of International conference on World Wild Web*, Madrid, Spain, 2009, pp. 791-800.

[41] S. Guo, B. Xiao, Y. Y. Yang, Y. Yang, Energy-efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing, *IEEE International Conference on Computer Communications*, San Francisco, CA, 2016, pp. 1-9.

Biographies

Xuhui Zhao received the Master's degree from Zhengzhou University, China. She is currently pursuing Ph.D. degree in Beijing University of Posts and Telecommunications, China. She is also a teacher at School of Information Engineering in Henan University of Science and Technology, China. Her research interests include mobile cloud computing and data mining.

Yan Shi received her Ph.D. degree from Beijing University of Posts and Telecommunications in 2007(BUPT). She is currently a member of the research staff of the State Key Laboratory of Networking and Switching Technology, BUPT. Her current research interests include vehicular networks and mobile computing, especially mobility management technology.

Shanzhi Chen received his Ph.D. degree from Beijing University of Posts and Telecommunications, China, in 1997. He joined Datang Telecom Technology & Industry Group in 1994, and has served as CTO since 2008. His current research interests include network architectures, 5G mobile communications, vehicular communication networks, Internet of Things.