A computation offloading method over big data for IoT-enabled cloud-edge computing

Xiaolong Xu, Qingxiang Liu, Yun Luo, Kai Peng, Xuyun Zhang, Shunmei Meng, Lianyong Qi

PII: S0167-739X(18)31977-0
DOI: https://doi.org/10.1016/j.future.2018.12.055
Reference: FUTURE 4676

To appear in: *Future Generation Computer Systems*

Received date : 16 August 2018
Revised date : 11 September 2018
Accepted date : 22 December 2018

Please cite this article as: X. Xu, Q. Liu, Y. Luo et al., A computation offloading method over big data for IoT-enabled cloud-edge computing, *Future Generation Computer Systems* (2019), https://doi.org/10.1016/j.future.2018.12.055

A Computation Offloading Method over Big Data for IoT-Enabled Cloud-Edge Computing

Xiaolong Xu[a,b,c], Qingxiang Liu[a,b], Yun Luo[d], Kai Peng[e], Xuyun Zhang[f], Shunmei Meng[g], Lianyong Qi[h,*]

[a] *School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, China*
[b] *Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, Nanjing, China*
[c] *State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China*
[d] *Faculty of Computer Science and Technology, Guizhou University, China*
[e] *Engineering Institute, Huaqiao University, China*
[f] *Department of Electrical and Computer Engineering, University of Auckland, New Zealand*
[g] *Department of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, China*
[h] *School of Information Science and Engineering, Qufu Normal University, China*

Abstract

The Internet of mobile things is a burgeoning technique that generates, stores and processes big real-time data to render rich services for mobile users. In order to mitigate conflicts between the resource limitation of mobile devices and users' demands of decreasing processing latency as well as prolonging battery life, it spurs a popular wave of offloading mobile applications for execution to centralized and decentralized data centers, such as cloud and edge servers. Due to the complexity and difference of mobile big data, arbitrarily offloading the mobile applications poses a remarkable challenge to optimizing the execution time and the energy consumption for mobile devices, despite the improved performance of Internet of Things (IoT) in cloud-edge computing. To address this challenge, we propose a computation offloading method, named COM, for IoT-enabled cloud-edge computing. Specificly, a system model is investigated,

*Corresponding author

Email addresses: njuxlxu@gmail.com (Xiaolong Xu), qingxiangliu737@gmail.com (Qingxiang Liu), yyyunll@gmail.com (Yun Luo), pkbupt@gmail.com (Kai Peng), xuyun.zhang@auckland.ac.nz (Xuyun Zhang), mengshunmei@njust.edu.cn (Shunmei Meng), lianyongqi@gmail.com (Lianyong Qi)

Preprint submitted to Future Generation Computer Systems *September 11, 2018*

including the execution time and energy consumption for mobile devices. Then dynamic schedules of data/control-constrained computing tasks are confirmed. In addition, NSGA-III (non-dominated sorting genetic algorithm III) is employed to address the multi-objective optimization problem of task offloading in cloud-edge computing. Finally, systematic experiments and comprehensive simulations are conducted to corroborate the efficiency of our proposed method.

Keywords: IoT, big data, cloud-edge computing, computation offloading, energy consumption

---

1. Introduction

1.1. Background

Internet of Things (IoT) has emerged as a popular paradigm providing internetworking of many objects and smart things, such as mobile devices and wearable devices [1][2]. Currently, the ever increasing mobile devices, embedded with radio frequency identification (RFID) and sensor technology, are connected to IoT via wireless networks, which integrates IoT with mobile computing. Due to the ubiquitous sensing, computing and integration, Internet of mobile things is used in a growing number of scenarios, e.g., healthcare system and catering business [3][4]. To render mobile users improved experiences and increase the service quality of mobile devices, the Internet of mobile things facilitates rich mobile applications, including measuring noise, recording location and capturing images [5].

Mobile devices in IoT sense the surroundings of mobile users and generate real-time big data, with useful information for supporting the mobile applications [6][7]. The big data is stored and processed to guarantee the efficiency and effectiveness of the mobile applications. However, the finite computation capacity and cache size of the mobile devices impede the wide usage of the mobile applications and cause tremendous amount of time for storing and processing the big data on the mobile devices [8][9]. Moreover, the energy consumption of the mobile devices increases, abbreviating the life of batteries and augmenting emissions of greenhouse gases.

To alleviate the resource limitation of the mobile devices and improve the performance of the generated mobile applications, cloud computing (CC) is a burgeoning computing scheme where the mobile applications are available to be offloaded to the centralized cloud data centers and the cloud manager provisions elastic and on-demand resources for executing the mobile applications[10][11]. In this way, the execution time of the mobile applications and the energy consumption of the mobile devices are reduced, which satisfies the mobile users' demands of shortening processing time and increasing battery life. Nevertheless, due to the cloud deployed distantly from the mobile devices, offloading the mobile applications to the remote cloud occupies substantial bandwidth of the core network, causing network congestion to a high extent. Furthermore, the mobile devices are connected to the cloud via Wide Area Network (WAN), and the bandwidth of offloading the mobile applications is low, which leads to high latency. Therefore, much time is depleted in the process of offloading the mobile applications to the cloud, causing immense offloading delay, especially for the data-intensive computing tasks [12][13].

Different from CC, edge computing (EC) pushes small data centers (such as cloudlets) with moderate resources, base stations and access points at the edge of radio access network, providing resource trusteeship services for the mobile devices under the coverage[14]. Cloudlets connect to mobile devices via Local Area Network (LAN), which is characterized by high bandwidth and low latency. Therefore, less time is consumed in the process of offloading mobile applications to the cloudlets, compared with that on the cloud, and the stress of core network is relieved. Hence, EC reduces offloading latency and makes network more efficient so that it provides a timesaving computing paradigm [15]. EC enables a hybrid computation offloading scheme, that is, mobile devices can offload the mobile applications to the cloudlet or to the cloud.

1.2. Motivation

To improve the performance of the mobile devices in IoT, cloudlets push cloud services to the network edge. Mobile applications are often formalized as workflows which contain some computing tasks with data/control dependencies. In cloud-edge computing, mobile devices in IoT are available to offload the computing tasks to the cloudlet or to the cloud for reducing the processing latency and prolonging the battery life of the mobile devices. However, arbitrarily offloading the computing tasks hardly optimizes the execution time and the energy consumption of the mobile devices, due to the moderate resources of the cloudlet and remote distance of the cloud. Therefore, it remains a challenge to optimize the execution time and the energy consumption of the mobile devices in the cloud-edge computing environment. To address the challenge, a hybrid computation offloading method for IoT-enabled cloud-edge computing is proposed.

1.3. Paper Contributions

In this paper, we make the following contributions.

- Analyze the execution time and the energy consumption of the mobile devices, and the computation offloading for IoT-enabled cloud-edge computing is defined as a multi-objective optimization problem.
- Confirm the dynamic schedules of the concurrent workflows in cloud-edge computing to select the optimal schedule strategy by using SAW (simple additive weighting) and MCDM (multiple criteria decision making).
- Adopt NSGA-III (non-dominated sorting genetic algorithm III) to address the multi-objective optimization problem of shortening the execution time and saving the energy consumption for each mobile device in IoT.
- Carry out comprehensive experiments and evaluations to validate the efficiency and effectiveness of COM.

The remainder of the paper is organized as follows. In Section 2, the problem formulation and the system model are proposed. In Section 3, a computation offloading method over big data for IoT-enabled cloud-edge computing is elaborated. Section 4 evaluates the proposed method. We discuss the related work in Section 5. Section 6 gives the conclusion and the future work.

2. System Model and Problem Fromulation

In this section, a system model in cloud-edge computing is proposed to evaluate the execution time and the energy consumption of the mobile devices. Key notations and descriptions are listed in Table 1.

Table 1: Key Notations and Descriptions

Notation	Descriptions
M	The number of mobile devices in IoT
V_m	The computing task set of the m-th workflow
ED_m	The dependency set of the m-th workflow
$d_{m,i}$	The input data of the computing task $v_{m,i}$ receives
$w_{m,i}$	The computation workload of the computing task $v_{m,i}$
X_m	The hybrid offloading strategies of the m-th workflow
$x_{m,i}$	The offloading strategy of the computing task $v_{m,i}$
$T^L(X_m)$	The offloading latency of the network
$T^e(X_m)$	The computing time in executing the m-th workflow
$T^t(X_m)$	The transmission time in executing the m-th workflow
A_i	The transmission strategies of two computing tasks
$T_m(X_m)$	The execution time of the m-th workflow
$E^L(X_m)$	The offloading energy consumption in executing the m-th workflow
$E^e(X_m)$	The computing energy consumption in executing the m-th workflow
$E^t(X_m)$	The transmission energy consumption in executing the m-th workflow
$E_m(X_m)$	The energy consumption for the m-th mobile device

2.1. Resource Model

The cloud-edge computing paradigm has the potential to satisfy the requirements of the execution time and the energy consumption for the mobile devices in IoT. Fig. 1 illustrates a system framework for IoT-enabled cloud-edge computing. In this framework, we consider a scenario where a cloudlet covers M mobile devices which are connected to a cloud deployed in the remote area. Each mobile application is formalized as a workflow, denoted as a directed acyclic graph (DAG). A workflow contains several data/control-constrained computing tasks. Let $DAG_m(V_m, ED_m)(m = \{1, 2, ..., M\})$ be the workflow running on the m-th mobile device, where $V_m = \{v_{m,i} | 1 \leq i \leq |V_m|\}$ represents the set of computing tasks in the m-th workflow and $ED_m = \{(v_{m,i}, v_{m,j}) | v_{m,i}, v_{m,j} \in V_m \bigwedge i \neq j\}$ describes the dependency between the computing tasks $v_{m,i}$ and $v_{m,j}$. Let the requirement-constrained data for processing of each computing task be a tuple, denoted as $(d_{m,i}, w_{m,i})$, where $d_{m,i}$ and $w_{m,i}$ reflect the input data the computing task $v_{m,i}$ receives from its precursor computing tasks and the computation workload for processing respectively. $pre(v_{m,i})$ represents the precursor computing tasks of $v_{m,i}$. Only all the computing tasks in $pre(v_{m,i})$ finish executions, can $v_{m,i}$ be executed. For example, there are a computing task set $\{v_1, v_2, v_3, v_4\}$ and a dependency set $\{(v_1, v_2), (v_1, v_3), (v_2, v_4)\}$. In this example, the precursor computing tasks of v_3 are v_1 and v_2, i.e., $pre(v_3) = \{v_1, v_2\}$.

In cloud-edge computing, the computing tasks in a workflow are available to be executed by the mobile device, the cloudlet or the cloud servers through computation offloading. X_m, a $|V_m|$-tuple, represents hybrid computation offloading strategies of the m-th workflow DAG_m. The element $x_{m,i}$ stands for the computation offloading strategy of the computing task $v_{m,i}$, which is measured as

$$x_{m,i} = \begin{cases} 0, & \text{if } v_{m,i} \text{ is executed in mobile device,} \\ 1, & \text{if } v_{m,i} \text{ is offloaded to the cloudlet,} \\ 2, & \text{if } v_{m,i} \text{ is offloaded to the cloud.} \end{cases} \quad (1)$$

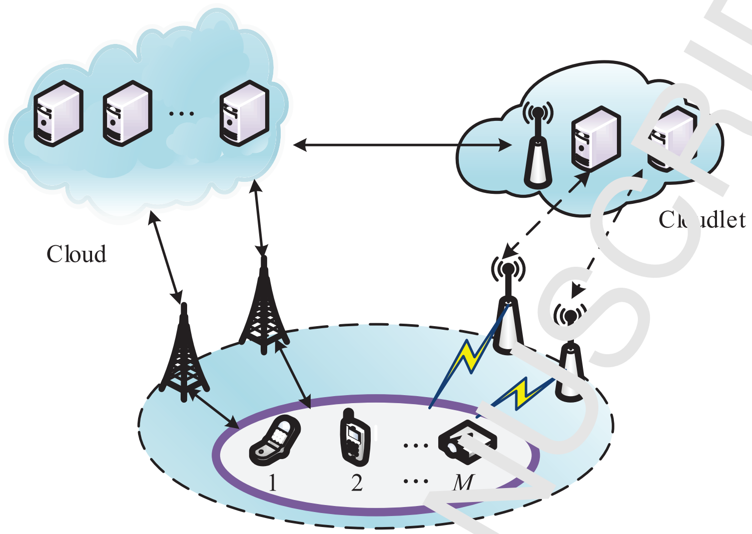


Figure 1: A system framework for IoT-enabled cloud-edge computing.

2.2. Execution Time Model

In the workflow execution, the latency of the network in computation offloading, the computing time of the computing tasks and the transmission time among the computing tasks are considered. Therefore, the execution time of DAG_m is divided into three categories, i.e., the offloading latency T^L, the computing time T^e, and the transmission time T^t.

For the computing task $v_{m,i}$, adopting the computation offloading strategy $x_{m,i}$, the offloading latency $T^L(x_{m,i})$ is calculated by

$$T^L(x_{m,i}) = \begin{cases} 0, & x_{m,i} = 0, \\ L_{LAN}, & x_{m,i} = 1, \\ L_{WAN}, & x_{m,i} = 2, \end{cases} \tag{2}$$

where L_{LAN} and L_{WAN} represent the latency of LAN and WAN respectively. Hence, the offloading latency of all the computing tasks in the m-th workflow, i.e., $T^L(X_m)$, is calculated by

$$T^L(X_m) = \sum_{x_{m,i} \in X_m} T^L(x_{m,i}). \tag{3}$$

In the execution of a computing task, the computing time is determined by the workload of the computing task and the computing power of the execution platform. Suppose the mobile devices in IoT transmit the offloading requests to the cloudlet according to the number of vacant virtual machines (VMs) in the cloudlet. If all the VMs have been instantiated, the cloudlet rejects the offloading requests. Instead of waiting for the available resources released from the occupied computing tasks deployed on the cloudlet, the mobile devices choose to execute these tasks or offload them to the cloud. Therefore, we neglect the queuing time for the execution of the computing tasks on the cloudlet in this paper. For the computing task $v_{m,i}$, the computing time $T^e(x_{m,i})$ is calculated by

$$T^e(x_{m,i}) = \begin{cases} \frac{w_{m,i}}{f_{local}}, & x_{m,i} = 0, \\ \frac{w_{m,i}}{f_{cl}}, & x_{m,i} = 1, \\ \frac{w_{m,i}}{f_c}, & x_{m,i} = 2, \end{cases} \tag{4}$$

where f_{local}, f_{cl} and f_c denote the computing power of the mobile devices, the cloudlet and the cloud respectively. Hence, the computing time for the execution of the m-th workflow is calculated by

$$T^e(X_m) = \sum_{x_{m,i} \in X_m} T^e(x_{m,i}). \tag{5}$$

The transmission time between two computing tasks with dependency relates to the offloading strategies of this two computing tasks. Let $A_i (i = 1, 2, 3)$ be the transmission strategy of the two computing tasks which is measured as

$$A_i = \begin{cases} \{(0,1),(1,0)\}, & i = 1, \\ \{(0,0),(1,1),(2,2)\}, & i = 2, \\ \{(0,2),(2,0),(1,2),(2,1)\}, & i = 3, \end{cases} \tag{6}$$

where $(x_{m,i}, x_{m,j}) \in A_1$ means the data is transmitted from the mobile device to the cloudlet, or conversely via LAN. If $(x_{m,i}, x_{m,j}) \in A_2$, the data is transmitted in the same computing environment. $(x_{m,i}, x_{m,j}) \in A_3$ refers to the data transmission between the mobile device and the cloud or between the cloudlet and the cloud via WAN. The transmission time between $v_{m,i}$ and $v_{m,j}$, denoted as $T^t(x_{m,i}, x_{m,j})$, is calculated by

$$T^t(x_{m,i}, x_{m,j}) = \begin{cases} \frac{d_{m,j}}{B_L}, & (x_{m,i}, x_{m,j}) \in A_1 \\ 0, & (x_{m,i}, x_{m,j}) \in A_2, \\ \frac{d_{m,i}}{B_W}, & (x_{m,i}, x_{m,j}) \in A_3, \end{cases} \tag{7}$$

where B_W and B_L represent the bandwidth of WAN and LAN respectively. The transmission time is determined by the workload of the transmission data and the bandwidth of the network. When the two computing tasks are executed in the same environment, the transmission time is neglected. The transmission time in the execution of the m-th workflow $T^t(X_m)$ is calculated by

$$T^t(X_m) = \sum_{(v_{m,i}, v_{m,j}) \in ED_m} T^t(x_{m,i}, x_{m,j}). \tag{8}$$

Let $T_m(X_m)$ be the execution time of the m-th workflow, which is calculated by

$$T_m(X_m) = T^L(X_m) + T^e(X_m) + T^t(X_m). \tag{9}$$

2.3. Energy Consumption Model for Mobile Devices

There is energy consumption for the mobile devices when offloading the computing tasks, executing the tasks and transmitting data among two computing tasks.

When the computing task $v_{m,i}$ adopts the offloading strategy $x_{m,i}$, the offloading energy consumption of the m-th mobile device, denoted as $E^L(x_{m,i})$, is calculated by

$$E^L(x_{m,i}) = T^L(x_{m,i}) \cdot p_I, \tag{10}$$

where p_I represents the idle power consumption of the mobile device. If a computing task is executed locally, there is no offloading energy consumption.

Therefore, the offloading energy consumption for the m-th mobile device in the execution of the m-th workflow is calculated by

$$E^L(X_m) = \sum_{x_{m,i} \in X_m} E^L(x_{m,i}). \tag{11}$$

For the computing task $v_{m,i}$, the computing energy consumption, represented as $E^e(x_{m,i})$, is calculated by

$$E^e(x_{m,i}) = \begin{cases} \frac{w_{m,i}}{f_{local}} p_A, & x_{m,i} = 0, \\ \frac{w_{m,i}}{f_{cl}} p_I, & x_{m,i} = 1, \\ \frac{w_{m,i}}{f_c} p_I, & x_{m,i} = 2, \end{cases} \tag{12}$$

where p_A denotes the active power consumption of the mobile device. When a computing task is implemented in the mobile device, the device becomes active. In addition, when the computing task is executed on the cloudlet or on the cloud, the mobile device is idle, but to maintain the successful execution of the workflow , there is still some certain power consumption. Thus, the computing energy consumption of the m-th mobile device in IoT is calculated by

$$E^e(X_m) = \sum_{x_{m,i} \in X_m} E^e(x_{m,i}). \tag{13}$$

$E^t(x_{m,i}, x_{m,j})$ represents the transmission energy consumption between $v_{m,i}$ and $v_{m,j}$, which is calculated by

$$E^t(x_{m,i}, x_{m,j}) = T^t(x_{m,i}, x_{m,j}) \cdot p_t, \tag{14}$$

where p_t denotes the transmission power consumption of the mobile device. Therefore, the transmission energy consumption for executing the m-th workflow is calculated by

$$E^t(X_m) = \sum_{(v_{m,i}, v_{m,j}) \in ED_m} E^t(x_{m,i}, x_{m,j}). \tag{15}$$

Suppose that each mobile device is equipped with a dynamic voltage and frequency system, which adjusts the voltage according to the computation load. Thus, p_I and p_t are lower than p_A.

Let $E_m(X_m)$ be the energy consumption of the m-th mobile device, and we can get:

$$E_m(X_m) = E^L(X_m) + E^e(X_m) + E^t(X_m). \tag{16}$$

2.4. Problem Formulation

In this paper, we intend to shorten the execution time given in (9) and save the energy consumption of each mobile device, presented in (16). The formalized problem is defined as

$$\min T_m(X_m), E_m(X_m), (\forall m \in \{1, 2, ..., M\}). \tag{17}$$

$$s.\ t. \quad \sum_{m=1}^{M} \mu_m \leq C, \tag{18}$$

$$\sum_{i=1}^{|V_m|} x_{m,i} = \mu_m (x_{m,i} = 1, \forall m \in \{1, 2, ...M\}), \tag{19}$$

$$T_m(pre(x_{m,i})) \leq T(pre(x_{m,i}) + x_{m,i})(i \leq |V_m|, m \leq M). \tag{20}$$

In this problem, C represents the maximum number of virtual machines (VMs) the cloudlet can instantiate and μ_m represents the number of instantiated VMs for executing the m-th workflow. The constraint presented in (18) describes that the aggregated computing resources of the instantiated VMs in a cloudlet are not over the computing capacity of the cloudlet. The constraint given in (19) indicates that each computing task offloaded to the cloudlet occupies one VM. The constraint elaborated in (20) ensures the precursor tasks of a computing task are implemented before the execution of it.

3. A Computation Offloading Method for IoT-Enabled Cloud-Edge Computing

In this section, we first confirm the dynamic schedules of concurrent workflows in cloud-edge computing. Then NSGA-III is utilized to find the global optimal solution. Finally, schedule evaluation is conducted based on SAW and MCDM to select the optimal solutions for the computing tasks in the same schedule.

3.1. Schedule Confirmation for Concurrent Workflows

In the execution of concurrent workflows, we separate the computing tasks in the workflows into three categories: the scheduled, the ready and the unready. Each time, we implement the computing tasks ready and suppose this process as a schedule. Consider after S schedules, the concurrent workflows finish executions. Let SKD=$\{skd_s|1\leq s\leq S\}$ represent the computing task sets for S schedules, where skd_s ($1\leq s \leq S$) represents the set of computing tasks executed in the s-th schedule. We consider each computing task in the workflow have similar computation load in this paper. It is depicted in Fig. 2 that there are two workflows, i.e., WF_1 and WF_2, for execution. In the first schedule, $v_{1,1}$ along with $v_{2,1}$, the root tasks of WF_1 and WF_2 are ready for execution. Thus, skd_1=$\{v_{1,1}, v_{1,2}\}$, skd_2=$\{v_{1,2}, v_{2,2}, v_{2,3}\}$, skd_3=$\{v_{1,3}, v_{1,4}, v_{2,4}\}$. After three schedules, the two workflows finish executions.

Algorithm 1 presents the confirmation of schedules for M concurrent workflows in cloud-edge computing. We input the workflow set, denoted as wf. U and V represent the set of scheduled computing tasks and the set of unscheduled computing tasks (Lines 2 and 3). If a computing task is the root of the remaining tasks in the same workflow, then the computing task is executed and we consider the M workflows simultaneously (Lines 5-10). Finally, the schedule times, and SKD are output.

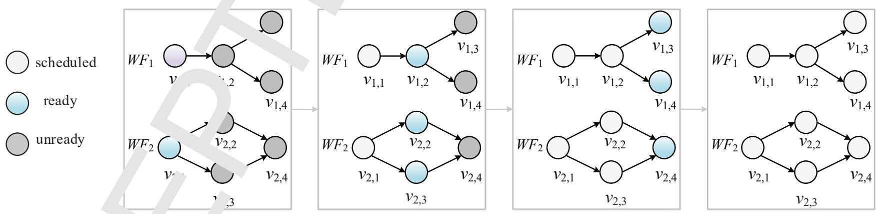


Figure 2: Dynamic schedules of two workflows with four computing tasks respectively.

3.2. Computation Offloading Method Using NSGA-III

In this subsection, the computation offloading of the computing tasks in each schedule is defined as a multi-objective optimization problem of shortening the

---

Algorithm 1 Schedule confirmation of concurrent workflows

Require: wf

Ensure: SKD, S

```
1:  s= 1
2:  U= ∅
3:  V= {v_{m,i}| 1≤ i≤ |V_m|, 1≤ i ≤ M}
4:  while U≠ V do
5:      for v_i∈ V do
6:          if pre(v_i)= ∅ then
7:              U= U∪{v_i}
8:              V= V-{v_i}
9:              skd_s= skd_s∪ {v_i}
10:         end if
11:         s= s+1
12:     end for
13: end while
14: S= s
15: return S, SKD
```

---

executing time and saving the energy consumption of mobile devices in IoT. NSGA-III is an efficient and accurate method for solving optimization problems with multiple objectives. Hence, NSGA-III is employed to address the multi-objective optimization problem given in (17).

We encode for the computation offloading strategies firstly. Then the fitness functions as well as constraints are discussed for the problem. Moreover, crossover and mutation operations are employed for the creation of new schedule solutions. Besides, the usual domination principle and the reference-point-based selection are adopted in the selection operation.

3.2.1. Encoding

We encode for the computation offloading strategies in this section. As is discussed in Section 2, each computing task has a computation offloading strategy. In the genetic algorithm (GA), a gene represents the computation offloading strategy of a computing task and the genes compromise a chromosome, reflecting a hybrid computation offloading of the computing tasks in the same schedule. Fig. 3 illustrates an example of computation offloading strategy encoding for the computing tasks in the first schedule. In this example, the chromosome is encoded in an array of integers (0, 1, 2).

x_1	x_2	x_3	$\cdots$	$x_{\|skd_1\|}$
1	2	0	$\ldots$	1

Figure 3: An encoding instance for the computing tasks in the first schedule.

3.2.2. Fitness Functions and Constraints

The fitness functions are utilized to judge whether a possible solution is optimal in GA. A chromosome is the offloading strategies of all the computing tasks of the same schedule and each chromosome is an individual, representing a solution of the multi-objective optimization problem. The fitness functions include two categories: the execution time and the energy consumption for each mobile device, presented in (9) and (16) respectively. The goal of the method is to find an optimal offloading strategy to minimize the two fitness functions for each mobile device, shown in (17). The fitness of a solution is to achieve the trade-offs between the $2M$ objectives.

In this method, we seek a hybrid offloading strategy of optimizing the execution time and the energy consumption for each mobile device. The constraints are given in (18), (19)and (20). NSGA-III performs well in addressing the optimization problem of multiple objectives with potential constraints.

The execution time is one fitness function. Algorithm 2 elaborates how we evaluate the execution time. In this algorithm, we input SKD and the offloading strategies, denoted as χ. We first calculate the offloading latency, the computing time and the transmission time for executing a computing task in each schedule (Lines 3-12) and then the total time for executing a workflow (Line 13). Finally, the time for executing each workflow is output in each schedule.

Algorithm 2 Execution time evaluation

Require: SKD, χ

Ensure: $T_m(X_m)$

1: **for** s= 1 to S **do**
2:     **for** m= 1 to M **do**
3:         **for** i= 1 to $|V_m|$ **do**
4:             Calculate $T^L(x_{m,i})$ by (2)
5:             Calculate $T^e(x_{m,i})$ by (3)
6:         **end for**
7:         Calculate $T^L(X_m)$ by (4)
8:         Calculate $T^e(X_m)$ by (5)
9:         **for** $(v_{m,i}, v_{m,j}) \in ED_m$ **do**
10:             Calculate $T^t(x_{m,i}, x_{m,j})$ by (7)
11:         **end for**
12:         Calculate $T^t(X_m)$ by (6)
13:         $T_m(X_m)= T^L(X_m)+ T^e(X_m)+ T^t(X_m)$
14:     **end for**
15: **end for**
16: **return** $T_m(X_m)$

The energy consumption of the mobile device is another fitness function. Algorithm 3 describes the process of evaluating the energy consumption. The inputs of the algorithm are the offloading latency, the computing time, the transmission time and the offloading strategies. The offloading energy consumption, the computing and the transmission energy consumption in executing each computing task is first calculated (Lines 2-11) and then the total energy consumption of the m-th mobile device in executing the workflow is obtained in each schedule

(Line 12). Finally, the outputs of the algorithm is the energy consumption for each mobile device in each schedule.

Algorithm 3 Energy consumption evaluation for mobile devices

Require: $T^L(x_{m,i})$, $T^e(x_{m,i})$, $T^t(x_{m,i}, x_{m,j})$, χ

Ensure: $E_m(X_m)$

1: **for** m= 1 to M **do**
2:     **for** i= 1 to $|V_m|$ **do**
3:         $E^L(x_{m,i})= T^L(x_{m,i})\cdot\, p_I$
4:         Calculate $E^e(x_{m,i})$ by (11)
5:     **end for**
6:     Calculate $E^L(X_m)$ by (12)
7:     Calculate $E^e(X_m)$ by (13)
8:     **for** $(v_{m,i},\, v_{m,j})\in ED_m$ **do**
9:         $E^t(x_{m,i}, x_{m,j})= T^t(x_{m,i}, x_{m,j})\cdot\, p$
10:     **end for**
11:     Calculate $E^t(X_m)$ by (15)
12:     $E_m(X_m)= E^L(X_m)+ E^e(X_m)+ E^t(X_m)$
13: **end for**
14: **return** $E_m(X_m)$

3.2.3. Initialization

In the subsection, the parameters of GA are determined, including the population size POP, the maximum of iteration I, the crossover possibility P_c, and the mutation possibility P_m.

Each chromosome represents the computation offloading strategies of the computing tasks in the same schedule. In addition, let the gene $c_{s,n}$ be the offloading strategy of the n-th computing task in the s-th schedule. In the s-th schedule, the chromosome is denoted as $C_{s,i}= (c_{s,1}, c_{s,2}, \ldots, c_{s,N})$ (i= 1, 2, ..., POP, N=$|skd_s|$).

3.2.4. Crossover and Mutation

In this paper, the standard single-point crossover operation is conducted to combine two chromosomes and generate two new individuals. Fig. 4 shows an example of crossover operation for two chromosomes in the first schedule. In this example, a crossover point is first determined, and then swap the genes around this point to create two new chromosomes.

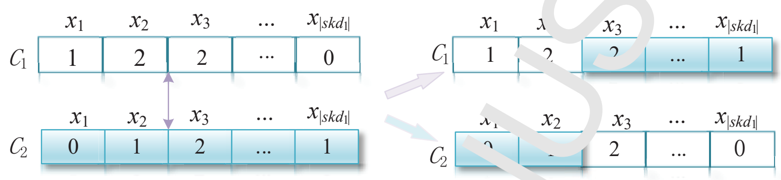


Figure 4: An example of crossover operation.

The mutation is to modify genes of the chromosomes in the hope of generating individuals with higher fitness values. Fig. 5 illustrates an example of the mutation operation in the first schedule. Each gene in a chromosome is changed with equal probability.

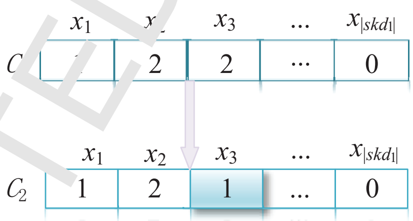


Figure 5: An example of mutation operation.

3.2.5. Selection for the Next Generation

In this phase, we aim at selecting the chromosomes for the next population to generate individuals with higher fitness values. Each chromosome represents a hybrid offloading strategy of the computing tasks in the same schedule. After crossover and mutation, the population size becomes $2POP$. Algorithms 2 and 3 are used to evaluate the values of two fitness functions for each workflow in a schedule. The solutions are sorted according to the $2M$ values to generate several non-dominated fronts using the usual domination principle.

In primary selection, we select one randomly from the solutions in the highest non-domination front each time to form the next generation until the number of the selected solutions is POP. Suppose the last added solution is in the l-th non-domination front. If all the solutions in the l-th front are included, then the selection finishes and the chosen solutions go into the next generation.

In further selection, consider z solutions in the l-th front are selected in primary selection. Then exclude the z solutions and further steps are conducted to make sure the z solutions in the l-th front should be included in the next generation.

We first normalize the $2M$ fitness values of each individual in the population. In the $2POP$ individuals, we search the minimum of the execution time and energy consumption for each mobile device, denoted as $T_m^*(X_m)(1 \leq m \leq M)$ and $E_m^*(X_m)(1 \leq m \leq M)$ respectively. Then the $2M$ values for $2POP$ individuals in the population are updated as

$$T'_m(X_m) = T_m(X_m) - T_m{}^*(X_m). \tag{21}$$

$$E'_m(X_m) = E_m(X_m) - E_m{}^*(X_m). \tag{22}$$

Let δ_T^m and δ_E^m represent the extreme values of the execution time and the energy consumption for the m-th mobile device respectively, which are calculated by

$$\delta_T^m = \max \frac{T'_m(X_m)}{W_{T_m}}. \tag{23}$$

$$\delta_E^m = \max \frac{E'_m(X_m)}{W_{E_m}}, \tag{24}$$

W_{T_m} and W_{E_m} in (23)(24) are the weight vectors of the two functions.

We consider each fitness function as an axis. In the hyperplane compromised by the $2M$ axes, the intercept of each axis is determined, denoted as α_T^m and α_E^m respectively for the m-th workflow. Then the $2M$ fitness values of each individual in the population are normalized as:

$$T''_m(X_m) = \frac{T'_m(X_m)}{\alpha_T^m}.. \tag{25}$$

$$E''_m(X_m) = \frac{E'}{\alpha_E^m}. \tag{26}$$

After the normalization, the values of the execution time and the energy consumption for each workflow are in the domain [0,1]. The solutions in the population has compromised a $2M$-dimensional hyperplane. Then the normalized solutions are associated with reference points. A set of reference points are scattered in the $2M$-dimensional hyperplane. The intercept of each axis is 1 and each of the axis is divided into g subsections. Then the number of the reference points, represented by θ, is calculated by

$$\theta = C_g^{2M+g-1}. \tag{27}$$

θ is approximately equal to the population size POP to make sure each normalized solution associates with one reference point nearly [16].

Sort the solutions in the l-th non-dominated front, according to the number of the reference points they associate with. Each time select one randomly from the solutions with maximum number of associated reference points. This process is repeated until all the z solutions have been selected.

The selection step is elaborated in Algorithm 4. In this algorithm, we input the t-th generation population (parent population) denoted as PP_t and the reference point set, denoted as R. The output is the $(t+1)$-th generation population (child population) PP_{t+1}. In this algorithm, we first calculate the execution time and the energy consumption of mobile devices by the Algorithms 2 and 3 respectively (Lines 2 and 3). Non-dominant sorting is conducted for individuals in the population through the usual domination principle (Line 5). Furthermore, we select the solutions primarily and judge whether all the solutions in the l-th front are included (Line 6). If not, we conduct further selection to determine the remaining z solutions in the l-th front for the next generation (Lines 8-12). Based on the selection algorithm, the POP solutions going into the next generation are selected.

Algorithm 4 Selection for the next generation

Require: PP_t, R

Ensure: PP_{t+1}

1: **for** $m = 1$ to M **do**
2: Calculate $T_m(X_m)$ by Algorithm 2
3: Calculate $E_m(X_m)$ by Algorithm 3
4: **end for**
5: Non-dominant sorting the POP solutions
6: Conduct Primary selection
7: **if** part of solutions in the l-th front are included **then**
8: Conduct further selection:
9: Normalize solutions for each workflow by (21-26)
10: Generate reference points under the constraints (27)
11: Associate the solutions with reference points
12: Select the remaining n solutions
13: **end if**
14: **return** PP_{t+1}

3.3. Schedule Selection Using SAW and MCDM

The proposed method aims at achieving trade-offs between optimizing the execution time and saving the energy consumption of mobile devices. In each population, there are POP chromosomes and each chromosome represents a hybrid computation offloading strategy of the computing tasks in a schedule. In addition, dynamic schedules of computing tasks are considered and to select relatively optimal schedule of each workflow, SAW and MCDM are employed.

The execution time is a negative criterion, that is, the higher the execution time is, the worse the solution becomes. Hence, the energy consumption of mobile devices is a negative criterion too. We normalize the execution time value in the m-th workflow execution as

$$V(T_m^j) = \begin{cases} \frac{{T_m}^{s,\max} - T_m(X_m^s)}{{T_m}^{s,\max} - {T_m}^{s,\min}}, & {T_m}^{s,\max} - {T_m}^{s,\min} \neq 0, \\ 1, & {T_m}^{s,\max} - {T_m}^{s,\min} = 0, \end{cases} \tag{28}$$

where $T_m^{s,max}$ and $T_m^{s,min}$ represent the maximum and minimum of the execution time in the s-th schedule of the m-th workflow. X_m^s represents the offloading strategies of computing tasks in the s-th schedule. Similarly, the energy consumption of m-th mobile device is normalized as

$$V(E_m^s) = \begin{cases} \frac{{E_m}^{s,\max} - E_m(X_m^s)}{{E_m}^{s,\max} - {E_m}^{s,\min}}, & {E_m}^{s,\max} - {E_m}^{s,\min} \neq 0, \\ 1, & {E_m}^{s,\max} - {E_m}^{s,\min} = 0, \end{cases} \tag{29}$$

where $E_m^{s,max}$ and $E_m^{s,min}$ represent the maximum and minimum of the energy consumption in the s-th schedule of the m-th workflow.

In addition, to calculate the utility value of each solution, the weight of each objective function requires determination. In this paper, we do an overall consideration of two objectives for each workflow. Therefore, the weights of the objectives are both $\frac{1}{2M}$. The utility value in the s-th schedule of the m-th workflow is calculated by

$$V(C_{s,i}) = \sum_{m=1}^{M} \frac{1}{2M} \cdot V(T_m^s) + \sum_{m=1}^{M} \frac{1}{2M} \cdot V(E_m^s) (1 \leq i \leq POP), \tag{30}$$

where $V(C_{s,i})$ represents the utility value of the i-th chromosome in the s-th schedule. Therefore, for each chromosome in the population, we have calculated the utility value of the same schedule. The optimal schedule solution, represented by $V(C_{s,i})$, is calculated by

$$V(C_s) = \max_{i=1}^{POP} V(C_{s,i}) \, (1 \leq s \leq S) \tag{31}$$

We can pick the optimal schedule with the maximum utility value in the POP chromosomes.

3.4. Method Overview

We aim at minimizing the execution time and the energy consumption of mobile devices in this paper. The computation offloading problem is defined as an optimization problem with multiple objectives and NSGA-III is adopted to obtain the global optimal offloading strategy. First, we confirm the dynamic schedules of concurrrent workflows. Then, the offloading strategies of computing tasks in each dynamic schedule are encoded as integers (0, 1, 2). In addition, the fitness functions and constraints are presented for the multi-objective optimization problem. Furthermore, the crossover and mutation operations are conducted to generate new individuals. The usual domination principle and reference-point-based selection in NSGA-III are adopted to pick out the individuals with best fitness for the next generation. Finally the schedule evaluation is proposed to select the optimal strategy for each schedule.

The overview of our proposed method is shown in Algorithm 5. We input the maximum iteration I and the initialized population X. The algorithm outputs the optimal computation offloading strategy in each schedule BX^s ($1\leq s\leq S$). Firstly, we obtain the dynamic schedules of the concurrent workflows and the schedule times (Line 1). By crossover and mutation, POP individuals are generated the population size becomes $2POP$ (Line 5). Then calculate the fitness functions of the $2POP$ solutions (Lines 6-8) and select the optimal individuals for the next generation (Line 10). For each schedule, the utility values are evaluated and the schedule strategy with the maximum utility value are picked out as the optimal schedule strategy (Lines 13 and 14). The process is repeated until the schedule iteration stops and finally the optimal strategies are the output.

4. Experimental Evaluation

In this section, we evaluate the performance of the proposed computation offloading method COM by comprehensive simulations and experiments.

Algorithm 5 Computation offloading method in cloud-edge computing

Require: I, X

Ensure: BX^s

```
Obtain SKD and S by Algorithm 2
for s=1 to S do do
    i= 1
    while i≤ I do
        Crossover and mutation operation
        for the individuals in the population do
            Calculate the execution time by Algorithm 2
            Calculate the device energy consumption by Algorithm 3
        end for
        Selection operation to ensure the child generation by Algorithm 4
        i= i+1
    end while
    Evaluate utility function in by (28-30)
    Pick out the optimal schedule strategy BX^s by (31)
end for
return X^n
```

4.1. Simulation Setup

In our stimulation, six mobile devices are under the coverage of the cloudlet and each of the six mobile devices has a mobile application for implementation respectively. i.e., WF_1, WF_2, WF_3, WF_4, WF_5, WF_6. The specific parameter settings in this experiment are given in Table 2.

The execution time and the energy consumption of the mobile devices in IoT are used to evaluate the performance of COM. To conduct the comparative analysis and validate whether COM is robust for computation offloading, the comparative methods are elaborated as follows:

- Benchmark: A workflow is implemented on the corresponding mobile device and the computing tasks are executed with control constraints. When the

Table 2: Parameter Settings

Parameter	Value
The idle power of mobile devices	0.002W
The active power of mobile devices	0.5W
The communication power of mobile devices	0.2W
The delay of LAN	0.5ms
The delay of WAN	30ms
The bandwidth of LAN	100kps
The bandwidth of WAN	50kps
The computing power of mobile devices	500MHZ
The computing power of the cloudlet	3000MHZ
The computing power of the cloud	5000MHZ
The number of VMs in the cloudlet	20

mobile device is overloaded, the computing tasks have to wait for execution until the resources are available. The process is repeated until all the computing tasks in the workflow have been executed.

• Cloudlet-oriented Computation Offloading Method (CLCOM): In this method, all the computing tasks in a workflow are offloaded to the cloudlet with control constraints. A VM on the cloudlet is instantiated if a computing task is offloaded to the cloudlet. If all the VMs on the cloudlet have been instantiated, the computing task has to wait for execution until the resources of the cloudlet are available. This process is repeated until all the computing tasks in the workflow have been executed.

• Cloud-oriented Computation Offloading Method (CCOM): In this method, the computing tasks in a workflow are all offloaded to the cloudlet with control constraints. A VM on the cloud is instantiated if a computing task is offloaded to the cloud. This process is repeated until all the computing tasks in the workflow have been executed.

The methods are implemented under the widespread used CloudSim framework on a personal computer with Intel Core i7-4720HQ 3.60GHz processors and 4GB RAM.

4.2. Performance Evaluation on COM

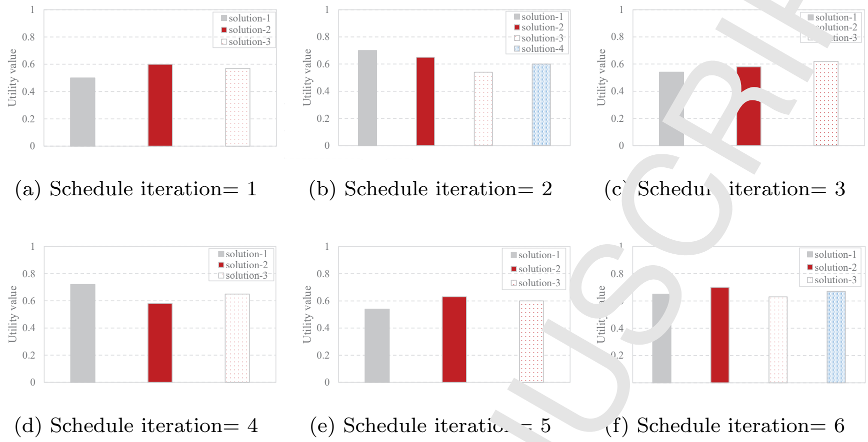
In this section, we evaluate the utility value in each dynamic schedule as well as the resource utilization for mobile devices. The corresponding results are shown in Fig. 6 and Fig. 7.

4.2.1. Evaluation of Utility values

In Section 3, SAW and MCDM are employed to select the relatively best solutions. We consider the dynamic schedule of the six concurrent workflows and after six schedules, the workflows finish their executions. For each schedule of workflows by COM, we calculate the utility values respectively in (25), (26) and (27). The most balanced schedule strategy is the solution with the maximum utility value. Six sub-figures in Fig. 6 illustrate the comparison of utility value in different schedules after the 1000^{th} iteration. It is explicit that the number of convergent solutions is 3, 4, 3, 3, 3 and 4 respectively corresponding to each schedule. For example, in Fig. 6a, the selected schedule strategy is solution-2, for the higher utility value than the other two. As is shown in Fig. 6f, after six schedules, the workflows are completed and the schedule strategy generated by solution-2 is the optimal one in the four solutions.

4.2.2. Evaluation of Resource Utilization

Resource utilization of the cloudlet is of great importance to the execution time and the energy consumption, and it is calculated according to the number of the VM instances on the cloudlet. In the dynamic schedules of workflows, we consider the resource utilization of the cloudlet changes with time instants, as is illustrated in Fig. 7. It is depicted that in the execution of the workflows, the resource utilization of the cloudlet is over 80% in most cases, which guarantees the good performance of COM. The reason why there is a sharp decrease of the



(a) Schedule iteration= 1 (b) Schedule iteration= 2 (c) Schedule iteration= 3

(d) Schedule iteration= 4 (e) Schedule iteration= 5 (f) Schedule iteration= 6

Figure 6: Comparison of utility value in different schedules by the generated solutions of COM.

resource utilization is that most of the computing tasks have finished execution.

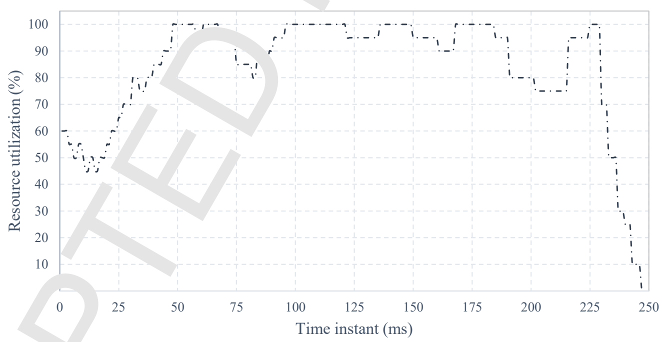


Figure 7: Resource utilization of the cloudlet at different time instants in the dynamic schedule of workflows by COM.

4.5. Comparison Analysis

In this section, we evaluate the performance of our proposed method and make comparisons with Benchmark, CLCOM and CCOM. The execution time and the energy consumption of mobile devices are two main metrics to assess the performance of the computation offloading methods. In addition, the power consumption of the mobile devices is used to compare the performance of the methods and we analyze the distribution of computing tasks in different methods. The corresponding results are illustrated in Fig. 8, Fig. 9, Fig. 10, and Fig. 11.

4.3.1. Comparison of Execution Time

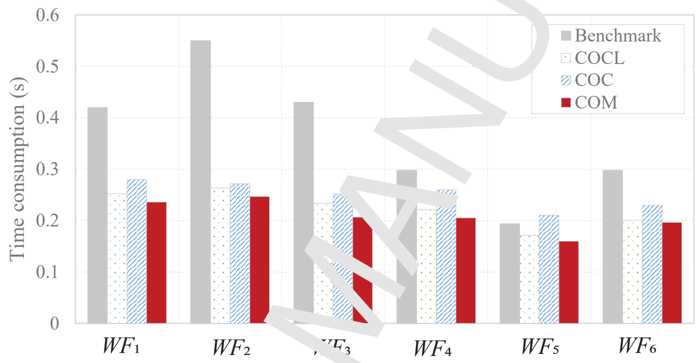


Figure 8: Comparison of execution time with different workflows by Benchmark, CLCOM, CCOM and COM.

The execution time consists of the offloading latency, the computing time and the transmission time. Fig. 8 shows the comparison of the execution time in executing six workflows by Benchmark, CLCOM, CCOM and COM. It is illustrated that our proposed method COM has the minimum execution time compared with the other methods.

In the Benchmark, all the computing tasks in a workflow are executed on the mobile device. Due to the resource limitation of the mobile device, the computing power is low, which makes much more time than the other three methods. In CCOM, a little more time is cost than in CLCOM, since the mobile devices connect to the cloudlet via LAN that has higher bandwidth and lower latency compared with WAN. Hence, less time is consumed when the workflow is executed on the cloudlet than on the cloud. Furthermore, resource capacity on the cloudlet is finite so that if VMs on the cloudlet are all instantiated, the remaining computing tasks in workflows requesting to be executed on the cloudlet have to wait until there are available resources in the cloudlet. However, in our proposed method COM, the hybrid offloading strategy is adopted, which makes the execution time in COM less than in CLCOM.

4.3.2. Comparison of Energy Consumption in Mobile Devices

The energy consumption of the mobile devices includes the offloading energy consumption, the computing energy consumption and the transmission energy consumption. Fig. 9 illustrates the comparison of the energy consumption for the mobile devices by Benchmark, CLCOM, CCOM and COM with different workflows. It is intuitive that the energy consumption of mobile devices by COM is less than that by the comparitive methods. The Mobile devices provide the processing energy and resources for the execution of workflows in Benchmark so that the energy consumption for the mobile devices is higher, compared with other methods. If a computing task is not implemented in the mobile device, the mobile device just needs to offload it to other computing platforms that supply the resources and the power for execution in CLCOM, CCOM and COM. Due to lower latency and higher bandwidth of LAN than those of WAN, the transmission energy consumption and the communication energy consumption in CLCOM are less than in CCOM. In COM, some of the computing tasks are executed in the mobile devices, so the offloading energy consumption is saved. Therefore, COM has further improvement in reducing the energy consumption for the mobile devices.

4.3.3. Comparison of power consumption in mobile devices

We obtain the power consumption according to the energy consumption in mobile devices and the execution time. Fig. 10 depicts the power consumption for mobile devices with different workflows by the four methods. The power consumption of the mobile device is an metric to measure the instant energy

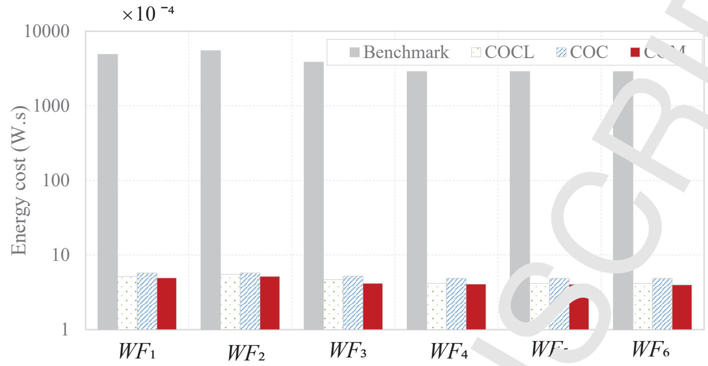


Figure 9: Comparison of energy consumption in mobile devices with different workflows by Benchmark, CLCOM, CCOM and COM.

consumption of mobile devices. If the power consumption is too high, there might be extreme energy consumption of mobile devices, which contributes to high energy consumption. From Fig. 10, the average power consumption of COM is a little lower than CLCOM and CCOM.

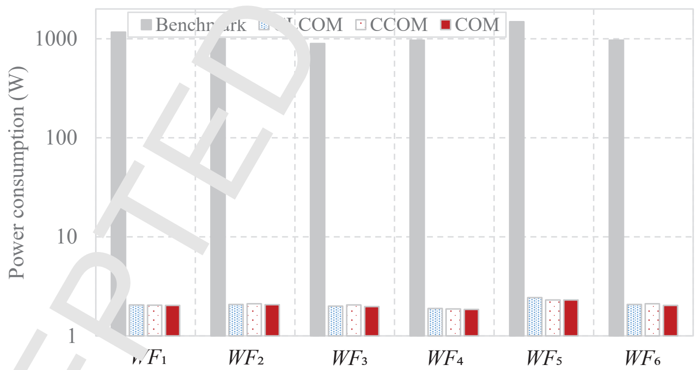
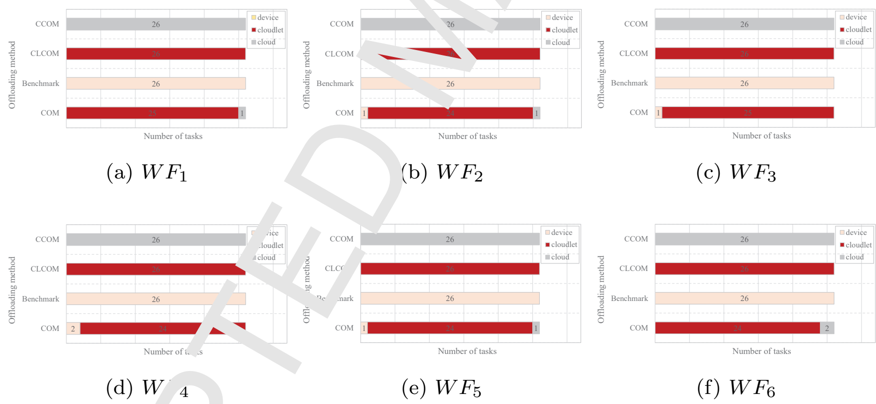


Figure 10: Comparison of power consumption with different workflows by Benchmark, CLCOM, CCOM and COM.

4.3.4. Comparison of Computing Task Distributions

In the proposed method COM, we consider a hybrid computation offloading strategy in cloud-edge computing, i.e., a computing task is implemented in mobile device, on the cloudlet or on the cloud. Fig. 11 illustrates the distribution of the computing tasks by different offloading methods respectively. It is explicit that most of the computing tasks are offloaded to the cloudlet in COM, due to the better performance of it in terms of reducing the execution time and decreasing the energy consumption for the mobile devices. However, since the moderate resource capacity of the cloudlet, if all the VMs on the cloudlet are instantiated, the computing task is executed in other computing platforms, instead of queuing in the cloudlet. Hence, few computing tasks are executed in the mobile devices or on the cloud, which guarantees the optimal execution time and energy consumption in mobile devices.



(a) WF_1 (b) WF_2 (c) WF_3

(d) WF_4 (e) WF_5 (f) WF_6

Figure 11: Comparison of computing task distributions in different schedules by the generated solutions of COM

5. Related Work

The IoT is a paradigm where everything around us can actively identify, connect, perceive and report the system on a global scale [17][18][19][20]. It enables interconnected smart devices to be used, monitored or configured for human beings [21][22]. Furthermore, the IoT is expected to play an important part in the construction of the next generation mobile communication services, which promotes more attractions focused on the Internet of mobile things [23].

The mobile devices in IoT generate big data from surroundings of mobile users to ensure the effectiveness and accuracy of the services that mobile applications provide [24][25][26][27]. In [28], Cai et al. the big data can be divided into four categories, i.e., Multisource High Heterogeneity Data, Huge Scale Dynamic Data, Low-Level with Weak Semantics Data and Inaccuracy Data. The generated IoT big data applications (IoTBDAs) are required to be equipped with capability of analyzing the data streams [29].

However, due to the resource limitation of mobile devices, users' demands of real-time processing and prolonged battery life are not guaranteed [30][31][32][33]. As a burgeoning technique, computation offloading to data centers such as cloud severs and cloudlets alleviates the problem. However, in spite of rich resource capacity, traditional clouds are deployed remotely from the mobile devices, which makes the offloading latency prolonged, especially for the computation-intensive tasks [34][35][36]. Compared with the cloud, EC is an emerging technology that aims at pushing applications and content close to the users to reduce latency, improve the quality of experience and ensure highly efficient network operation and service delivery, which has the potential to address the concerns of response time requirement, battery life constraint and bandwidth cost saving [37][38][39][40][41].

Considering the increasing mobile applications, to process the big data from the mobile devices in IoT, designing an EC framework is of great significance [42]. In [43], Li et al. proposed a novel edge computing for IoT (ECIoT) architecture, and investigated radio resource and computational resource management in ECIoT to enhance the system performance. In [44], Amjad et al. proposed a cognitive edge-computing based framework solution to achieve an efficient usage of the distributed resources with the dynamic extensive computing facilities of the cloudlet for end-users. The EC scheme reduces the execution time and saves the energy consumption of the mobile devices in IoT, thus improving the quality of user experience. However, on the other hand, the computing resources on the cloudlet are finite, which calls for the resource coordination between the cloudlet and the cloud. In [45], Jeong et al. discussed a system of mobile cloud computing (MCC) based on unmanned aerial vehicles (UAVs) to reduce the mobile energy consumption through computation offloading. In the system, joint optimization of bit allocation and trajectory of cloudlet was proposed. Jin et al. proposed an incentive-compatible mechanism (ICAM) to distribute cloudlets based on the demand of mobile services [46].

As momentous part of EC, computation offloading promises the decreased execution time and energy consumption of mobile devices in IoT. In the gross, computation offloading purports offloading the workloads to cloud servers or cloudlets. In [47], Roy Et al. proposed an application-aware cloudlet selection strategy to reduce the energy consumption of the mobile terminals and the execution latency of the mobile applications. With this strategy, the computing tasks are offloaded to the suitable cloudlet according to the application type in multi-cloudlet scenario. Code offloading for image processing tasks in mobile applications was investigated to ameliorate the performance and energy consumption [48]. In [49], Alasmari et al. proposed a Markov Decision Process (MDP) to seek a hybrid offloading strategy in mobile devices, the edge and the cloud, while optimizing the execution time and the energy consumption. Dinh et al. researched offloading from a mobile device to several edges. In such scenario, task allocation and central process unit frequency of the mobile device are optimized to reduce the execution latency and energy consumption of the mobile device [50]. Zhang et al. proposed a joint computation offloading and resource optimization in MEC. In such scheme, computation offloading strategy was studied to reduce the energy consumption and execution time [51].

However, to the best of our knowledge, few of the existing works have investigated the multi-objective optimization of computation offloading problems for IoT-enabled cloud-edge computing. With the observations above, it is still a challenge to realize the goals of reducing the execution time and saving the energy consumption for the mobile devices in IoT. In view of this challenge, a computation offloading method in cloud-edge computing environment is proposed this paper.

6. Conclusion and Future Work

Nowadays, Internet of mobile things has emerged as a popular technology for bringing about rich mobile applications. With the development of the technology, the complexity and scales of the big data for process increase, which has conflicts with the resource limitation of mobile devices. EC paradigm alleviates the problem to a great deal by offloading computing tasks to the cloud or to the cloudlet. In a bid to realize multi-objective optimization of reducing the execution time and saving the energy consumption for mobile devices, a computation offloading method, named COM, is proposed in this paper. Firstly, we analyzed the dynamic schedules of concurrent workflows and then NSGA-III is exploited to address the multi-objective optimization problem. Furthermore, extensive experiments and evaluations are conducted to affirm the proposed method COM performs well in solving the optimization problem.

For future work, we will adjust and extend the proposed method in a real-world scenario of IoT. In addition, we will crystallize different time requirements of the workflows for execution, trying to find an offloading strategy to achieve the maximum energy consumption savings of the mobile devices.

7. Acknowledgement

This research is supported by the National Science Foundation of China under grant no. 61702277, no. 61872219, no. 6177228 and no. 616722763. This work is also supported by The Startup Foundation for Introducing Talent of NUIST, the open project from the State Key Laboratory for Novel Software Technology, Nanjing University, under grant no. KFKT2017B04, the Priority Academic Program Development of Jiangsu Higher Education Institutions (PAPD) fund, and Jiangsu Collaborative Innovation Center on Atmospheric Environment and Equipment Technology (CICAEET).

References

[1] M. Bahrami, A. Khan, M. Singhal, An energy efficient data privacy scheme for iot devices in mobile cloud computing, in: IEEE International Conference on Mobile Services, 2016, pp. 190–195.

[2] R. H. Jhaveri, N. M. Patel, Y. Zhong, A. K. Sangaiah, Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad-hoc networks in industrial iot, IEEE Access (2018) 1–1.

[3] M. A. Razzaque, M. Milojevic-Jevric, A. Palade, S. Clarke, Middleware for internet of things: A survey, IEEE Internet of Things Journal 3 (1) (2017) 70–95.

[4] P. Kamalinejad, C. Mahapatra, Z. Sheng, S. Mirabbasi, V. C. M. Leung, Y. L. Guan, Wireless energy harvesting for the internet of things, IEEE Communications Magazine 53 (6) (2015) 102–108.

[5] S. Abolfazli, Z. Sanaei, M. Alizadeh, A. Gani, F. Xia, An experimental analysis on cloud-based mobile augmentation in mobile cloud computing, IEEE Transactions on Consumer Electronics 60 (1) (2014) 146–154.

[6] X. Wang, W. Wang, L. T. Yang, S. Liao, D. Yin, M. J. Deen, A distributed hosvd method with its incremental computation for big data in cyber-physical-social systems, IEEE Transactions on Computational Social Systems PP (99) (2018) 1–12.

[7] C. Wu, E. Zapevalova, Y. Chen, F. Li, Time optimization of multiple knowledge transfers in the big data environment, Computers, Materials & Continua 54 (3) (2018) 269–285.

[8] X. Chen, Q. Shi, L. Yang, J. Xu, Thriftyedge: Resource-efficient edge computing for intelligent iot applications, IEEE Network 32 (1) (2018) 61–65.

[9] X. Wang, L. T. Yang, H. Liu, M. J. Deen, A big data-as-a-service framework: State-of-the-art and perspectives, IEEE Transactions on Big Data PP (99) (2017) 1–1.

[10] X. Wang, L. T. Yang, X. Xie, J. Jin, M. J. Deen, A cloud-edge computing framework for cyber-physical-social services, IEEE Communications Magazine 55 (11) (2017) 80–85.

[11] Y. Xu, L. Qi, W. Dou, J. Yu, Privacy-preserving and scalable service recommendation based on simhash in a distributed cloud environment, Complexity 2017.

[12] L. Qi, S. Meng, X. Zhang, R. Wang, X. Xu, Z. Zhou, W. Dou, An exception handling approach for privacy-preserving service recommendation failure in a cloud environment, Sensors 18 (7) (2018) 2037.

[13] F. Wu, X. Zhang, W. Yao, Z. Zheng, L. Xiang, W. Li, An advanced quantum-resistant signature scheme for cloud based on eisenstein ring, Computers, Materials & Continua 56 (1) (2018) 19–34.

[14] N. Ansari, X. Sun, Mobile edge computing empowers internet of things, IEICE Transactions on Communications 101 (3) (2018) 604–619.

[15] M. Bouet, V. Conan, Mobile edge computing resources optimization: a geo-clustering approach, IEEE Transactions on Network and Service Management 15 (2) (2018) 787–796.

[16] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601.

[17] B. R. Ray, M. U. Chowdhury, J. H. Abawajy, Secure object tracking protocol for the internet of things, IEEE Internet of Things Journal 3 (4) (2016) 544–553.

[18] J. Kaur, K. Kaur, A fuzzy approach for an iot-based automated employee performance appraisal, Computers, Materials & Continua 53 (1) (2017) 23–36.

[19] Z. Pan, J. Lei, Y. Zhang, F. L. Wang, Adaptive fractional-pixel motion estimation skipped algorithm for efficient hevc motion estimation, ACM Transactions on Multimedia Computing, Communications, and Applications 14 (1) (2018) 12.

[20] X. Guo, L. Chu, X. Sun, Accurate localization of multiple sources using semidefinite programming based on incomplete range matrix, IEEE Sensors Journal 16 (13) (2016) 5319–5324.

[21] E. Rubio-Drosdov, D. Díaz-Sánchez, F. Almenárez, P. Arias-Cabarcos, A. Marín, Seamless human-device interaction in the internet of things, IEEE Transactions on Consumer Electronics 63 (4) (2017) 490–498.

[22] J. Shen, C. Wang, T. Li, X. Chen, X. Huang, Z.-H. Zhan, Secure data uploading scheme for a smart home system, Information Sciences 453 (2018) 186–197.

[23] M. J. Farooq, Q. Zhu, A multi-layer feedback system approach to resilient connectivity of remotely deployed mobile internet of things, IEEE Transactions on Cognitive Communications and Networking 2018.

[24] J. Cui, Y. Zhang, Z. Cai, A. Liu, Y. Li, Securing display path for security-sensitive applications on mobile devices, CMC Comput. Mater. Contin 55 (2018) 17–35.

[25] R. Cheng, R. Xu, X. Tang, V. S. Sheng, C. Cai, An abnormal network flow feature sequence prediction approach for ddos attacks detection in big data environment, Computers, Materials & Continua 55 (1) (2018) 095–095.

[26] C. Wang, J. Shen, Q. Liu, Y. Ren, T. Li, A novel security scheme based on instant encrypted transmission for internet of things, Security and Communication Networks 2018.

[27] Q. Zhang, L. T. Yang, Z. Chen, P. Li, A survey on deep learning for big data, Information Fusion 42 (2018) 146–157.

[28] H. Cai, B. Xu, L. Jiang, A. V. Vasilakos, Iot-based big data storage systems in cloud computing: Perspectives and challenges, IEEE Internet of Things Journal 4 (1) (2017) 75–87.

[29] R. Ranjan, D. Thakker, A. Haller, R. Buyya, A note on exploration of iot generated big data using semantics (2017).

[30] L. Yang, H. Zhang, M. Li, J. Guo, H. Ji, Mobile edge computing empowered energy efficient task offloading in 5g, IEEE Transactions on Vehicular Technology 2018 67 (7).

[31] X. Wu, C. Zhang, R. Zhang, Y. Wang, J. Cui, A distributed intrusion detection model via nondestructive partitioning and balanced allocation for big data, Computers, Materials & Continua 56 (1) (2018) 061–072.

[32] L. Xiong, Y. Shi, On the privacy-preserving outsourcing scheme of reversible data hiding over encrypted image data in cloud computing, Computers, Materials & Continua 2018.

[33] L. Yang, Z. Han, Z. Huang, J. Ma, A remotely keyed file encryption scheme under mobile cloud computing, Journal of Network and Computer Applications 106 (2018) 90–99.

[34] Z. Cai, H. Yan, P. Li, Z.-a. Huang, C. Gao, Towards secure and flexible ehr sharing in mobile health cloud under static assumptions, Cluster Computing 20 (3) (2017) 2415–2422.

[35] J. Shen, Z. Gui, S. Ji, J. Shen, H. Tan, Y. Tang, Cloud-aided lightweight certificateless authentication protocol with anonymity for wireless body area networks, Journal of Network and Computer Applications 106 (2018) 117–123.

[36] J. Zhang, N. Xie, X. Zhang, K. Yue, W. Li, D. Kumar, Machine learning based resource allocation of cloud computing in auction, Computers, Materials & Continua 56 (1) (2018) 123–135.

[37] B. P. Rimal, D. P. Van, M. Maier, Cloudlet enhanced fiber-wireless access networks for mobile-edge computing, IEEE Transactions on Wireless Communications 16 (6) (2017) 3601–3618.

[38] M. V. Barbera, S. Kosta, A. Mei, J. Stefa, To offload or not to offload? the bandwidth and energy costs of mobile cloud computing, in: INFOCOM, 2013 Proceedings IEEE, 2013, pp. 1285–1293.

[39] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646.

[40] W. Quan, K. Wang, Y. Liu, N. Cheng, H. Zhang, X. S. Shen, Software-defined collaborative offloading for heterogeneous vehicular networks, Wireless Communications and Mobile Computing 2018.

[41] K. Wang, H. Yin, W. Quan, G. Min, Enabling collaborative edge computing for software defined vehicular networks, IEEE Network PP (99) (2018) 1–6.

[42] W. Quan, Y. Liu, H. Zhang, S. Yu, Enhancing crowd collaborations for software defined vehicular networks, IEEE Communications Magazine 55 (8) (2017) 80–86.

[43] S. Li, N. Zhang, S. Lin, L. Kong, A. Katangur, M. K. Khan, M. Ni, G. Zhu, Joint admission control and resource allocation in edge computing for internet of things, IEEE Network 32 (1) (2018) 72–79.

[44] A. Amjad, F. Rabby, S. Sadia, M. Patwary, E. Benkhelifa, Cognitive edge computing based resource allocation framework for internet of things, in: Fog and Mobile Edge Computing (FMEC), 2017 Second International Conference on, IEEE, 2017, pp. 194–200.

[45] S. Jeong, O. Simeone, J. Kang, Mobile edge computing via a uav-mounted cloudlet: Optimization of bit allocation and path planning, IEEE Transactions on Vehicular Technology 67 (3) (2018) 2049–2063.

[46] A.-L. Jin, W. Song, W. Zhuang, Auction-based resource allocation for sharing cloudlets in mobile cloud computing, IEEE Transactions on Emerging Topics in Computing 6 (1) (2018) 45–57.

[47] D. G. Roy, D. De, A. Mukherjee, R. Buyya, Application-aware cloudlet selection for computation offloading in multi-cloudlet environment, The Journal of Supercomputing 73 (4) (2017) 1672–1690.

[48] G. Valenzuela, A. Neyem, J. I. Benedetto, J. Navon, D. Sanabria, J. A. Karmy, F. Balbontin, Towards native code offloading platforms for image processing in mobile applications: a case study, in: Proceedings of the 4th International Conference on Mobile Software Engineering and Systems, IEEE Press, 2017, pp. 221–222.

[49] K. R. Alasmari, R. C. Green, M. Alam, Mobile edge offloading using markov decision processes, in: International Conference on Edge Computing, Springer, 2018, pp. 80–90.

[50] T. Q. Dinh, J. Tang, Q. D. La, T. Q. Quek, Offloading in mobile edge computing: Task allocation and computational frequency scaling, IEEE Transactions on Communications 65 (8) (2017) 3571–3584.

[51] J. Zhang, W. Xia, F. Yan, L. Shen, Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing, IEEE Access 6 (2018) 19324–19337.

1. Xiaolong Xu received his Ph.D. degree from the Nanjing University, China, in 2016. He worked as a research scholar at Michigan State University, USA, from Apr. 2017 to May 2018. He is currently an assistant professor with the school of computer and software, Nanjing University of Information Science and Technology. He has published more than 40 peer review papers in international journals and conferences. His research interests include mobile computing, edge computing, IoT, cloud computing and big data.

2. Qingxiang Liu is currently working towards his B.S. degree in Computer Science and Technology at School of Computer and Software in Nanjing University of Information Science and Technology. His research interests include Big Data and Mobile Cloud Computing.

3. Yun Luo received her Master and Bachelor degrees in Computer Science and Technology from Guizhou University in 2008 and 2011, respectively. She worked as an ICT engineer in the ICT Department of Guizhou Air Traffic Management Sub-bureau of Civil Aviation Administration of China from 2011 to 2015. At present, she works as a part-time associate research assistant in the Database Technology Research Laboratory in Guizhou University. She has participated in several IT related research and development projects, and published several research papers in international conferences. Her research interests include data warehouse, database and data mining, big data analytics and cloud computing.

4. Kai Peng completed his Ph.D. degree in State Key Laboratory of networking and switching technology, Beijing University of Posts and Telecommunications in July, 2014. He has been awarded a scholarship under the Stated Scholarship Fund to work as a visiting faculty (2017 ~2018) at the Department of Electrical and Computer Engineering, the University of British Columbia. He is currently an Assistant Professor in College of Engineering at Huaqiao University. He has served or is currently serving as a main organizer of international conferences and workshops such as SCS 2017, PIMRC 2017, CPSCom 2018, ICMU 2018 and so forth. Dr. Kai Peng has co-authored more than 15 journal/conference papers. He is a member of IEEE, CCF.

5. Xuyun Zhang is a lecturer in the Department of Electrical & Computer Engineering at the University of Auckland, New Zealand. Prior to his current appointment, he worked as a postdoctoral fellow in the Machine Learning Research Group of NICTA (currently Data61, CSIRO) in Australia. He received his PhD degree from University of Technology, Sydney (UTS, Australia) in 2014, as well as his ME and BS degrees in Computer Science from Nanjing University (China) in 2011 and 2008, respectively. His primary research interests include IoT and smart cities, big data, cloud computing, scalable machine learning & data mining, data privacy & security, and Web service technology.

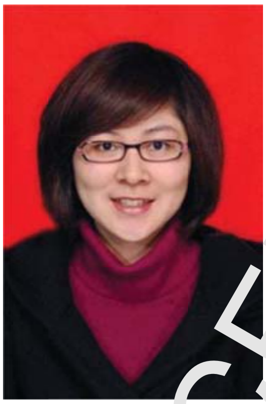
6. Shunmei Meng received her PhD degree in Department of Computer Science and Technology from Nanjing University, China, in 2016. Now, she is an assistant professor of School of Computer Science and Technology, Nanjing University of Science and Technology, Nanjing, China. She has published papers in international journals and international conferences. Her research interests include recommender systems, service computing, and cloud computing.

7. Lianyong Qi received his PhD degree in Department of Computer Science and Technology from Nanjing University, China, in 2011. Now, he is an associate professor of the School of Information Science and Engineering, Chinese Academy of Education Big Data, Qufu Normal University, China. He has already published more than 30 papers including JSAC, TCC, TBD, FGCS, JCSS, CCPE. His research interests include recommender systems and services computing.

1. Xiaolong Xu

2. Qingxiang Liu

3. Yun Luo



4. Kai Peng

5. Xuyun Zhang

6. Shunmei Meng

7. Lianyong Qi

1. Analyze the dynamic schedules according to the data or control dependencies of the computing tasks.
2. Adopt NSGA-III to address the multi-objective optimization problem in IoT.
3. Select the optimal schedule strategy by leveraging SAW and MCDM techniques.