A Multi-objective Computation Offloading Method for Hybrid Workflow Applications in Mobile Edge Computing

Kai Peng[1,2(✉)], Bohai Zhao[1], Xingda Qian[1], Xiaolong Xu[3], Lixin Zheng[2], and Victor C. M. Leung[4]

[1] College of Engineering, Huaqiao University, Quanzhou, China
pkbupt@gmail.com
[2] Fujian Provincial Academic Engineering Research Center in Industrial Intellectual Techniques and Systems, Quanzhou, China
[3] Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing 210094, People's Republic of China
[4] Department of Electrical and Computer Engineering, The University of British Columbia, Vancouver, BC V6T 1Z4, Canada

Abstract. Computation offloading has become a promising method to overcome intrinsic defects of portable smart devices, such as low operating speed and low battery capacity. However, it is a challenge to design an optimized strategy as the edge server is resource-constrained and the workflow application has timing constraints. In this paper, we investigated the hybrid workflow application computation offloading issue, which further increases the difficulty. According to the analysis of theory and consideration of time consumption and energy consumption, we establish a multi-objective optimization model to solve the issue. Furthermore, we propose a method based on particle swarm optimization algorithm for multi-objective computation offloading to get the optimal strategy for tasks offloading, which is suitable for all the hybrid workflow applications. Finally, extensive experiments have verified the effectiveness and efficiency of our proposed method.

Keywords: Mobile edge computing · Hybrid workflow applications · Multi-objective · Time consumption · Energy consumption

1 Introduction

With the development of cloud computing and big data, mobile devices are becoming an essential part of people's daily life [1,2]. People can obtain all the services they need from cloud computing with the help of mobile devices. Nevertheless, the distance between the remote cloud and users is quite far, which may reduce the quality of service [3,4]. Fortunately, there is another computing platform named mobile edge computing (MEC), which can push computation and storage closer to mobile users [5,6]. Computation offloading of MEC can solve the resource limitation problem of mobile devices effectively. Namely, offloading some applications from local to the edge servers in MEC may be more energy efficient [7,8].

In this paper, we mainly focus on the computation offloading of workflow applications(WAs) in MEC. Although many studies investigate the computation offloading of WAs in mobile cloud computing (MCC) well [9–13], the solutions in MCC cannot be used for the issue in MEC directly due to the different architectures of MCC and MEC. The MEC has a three-tier architecture, the core of which is the edge server [14]. Besides, we mainly focus on the cloudlet-based MEC in this study [15]. The Edge server can significantly reduce the delay for users to access the remote cloud, but the computing resources of the cloudlet are limited. Therefore, the computation offloading strategy has a significant impact on the completion time of the WA and the energy consumption of the mobile device (MD).

In this paper, the hybrid WAs computation offloading in MEC is well investigated. Hybrid applications consist of several general WAs and several time-constrained WAs. Our primary objective is finding optimization computation offloading strategies to minimize the time consumption and the energy consumption of each WA while meeting the time constraints of each WA that was given in advance. Our primary contributions are summarized as follows.

1) We investigated the hybrid WAs computation offloading issue in MEC.

2) The hybrid WAs computation offloading is molded as a multi-objective optimization problem, both energy consumption and energy consumption are taken into consideration.

3) A multi-objective computation offloading method for hybrid WAs based on improved particle swarm optimization is designed to minimize time consumption and energy consumption for each WA while satisfying the deadline requirements of the WAs.

4) Compared to other methods, the experimental results verify that our proposed method is effective.

The rest of this paper is organized as follows. Section 2 introduces related work. The system model and problem formulation are introduced in Sect. 3. Section 4 gives the multi-objective offloading algorithm for the hybrid WAs method. Section 5 introduces the experimental results and discussion. Section 6 describes the conclusion of this paper and the future work.

2 Related Work

Computation offloading has been well studied in [9–11]. Jia et al. [9] studied the migration of linear topology tasks and parallel topology tasks in the MCC environment and proposed an optimal linear task offloading strategy and heuristic parallel task migration algorithm based on the strategy of load balancing. Deng et al. [10] proposed a computational load offloading strategy for WAs based on a genetic algorithm. Based on their algorithm, some tasks of the WA are offloaded to the remote cloud to execute so as to achieve the optimization of execution time of WA and energy consumption of devices. Zhou et al. [11] consider the delay transmission mechanism and then propose a multi-objective workflow scheduling algorithm. They incorporate the delay transmission mechanism in the WA scheduling process, which can effectively optimize the energy consumption and completion time of WA at the same time.

As a result of the different architecture between MCC and MEC, the computation offloading of WAs in MCC cannot be used for the issue in MEC.

Computation offloading of general applications has been studied in [18]. In terms of using augmented reality applications in the environment of MEC, the authors proposed a corresponding algorithm for computation offloading. Furthermore, they proposed a multi-user computing offloading algorithm for the applications with high latency requirements and multi-user participation. Li et al. [19] try their utmost to minimize the computation latency of each task by proposing a migration algorithm. In this way, the applications are divided into many parts and are migrated to multiple cloudlets. But, they merely consider the latency optimization.

Zhang et al. [20] proposed an offloading strategy which reduced the energy of home automation applications. In order to reduce the WAs$'$ total energy consumption within the constrained deadline, the MDs are made scheduled using the particle swarm optimization. Nevertheless, their method focuses on the optimization of energy consumption.

In this paper, we focus on the multi-objective optimization for hybrid WAs in MEC. Both energy consumption and time consumption are taken into consideration.

3 The System Model and Problem Formulation

In this section, the architecture of MEC is introduced. As shown in Fig. 1, we can see that MEC has a three-tire architecture which consists of mobile users, MEC servers, and remote cloud. In addition, there are some applications like WA in mobile devices that should be executed within given constrained time. In addition, these applications can be processed locally or can be offloaded to the cloud via WAN or cloudlet via LAN according to the strategy to reduce energy consumption or execution time, or both of them.

3.1 Basic Mode

In this section, the referred variable symbols are introduced. N represents computation task. N_f represents the number of the f_{th} WA's computation tasks. The workflow is used to model the mobile application, denoted as $W = \{w_1, w_2, \ldots, w_f\}$. In this paper, the WAs include scheduled and unscheduled WA which are denoted as $W_f(V, \xi)$. The set of computation tasks are represented by $V = \{v_{1,f}, v_{2,f}, \ldots, v_{i,f}\}$ and $\xi = \{r(v_{i,f}, v_{j,f}) | v_{i,f}, v_{j,f} \in V\}$ represents the dependencies between tasks in the certain workflow. S represents the

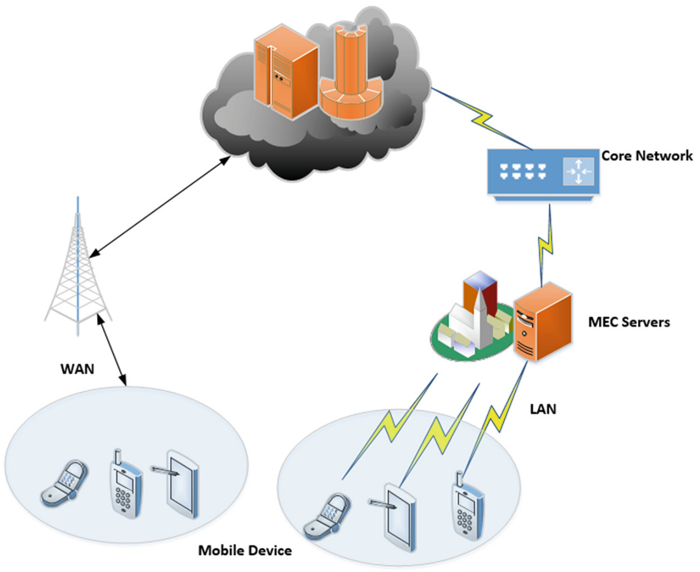


Fig. 1. The architecture of MEC

set of offloading strategies. $S = \{S_{i,f} | i = 1, 2, \ldots, N, f = 1, 2, \ldots, F\}$, where $S_{i,f} \in \{1, 2, 3\}$. $S_{i,f} = 1$ means that the $v_{i,f}$ is executed on the local network, $S_{i,f} = 2$ denotes the task is offloaded and computed by the cloudlet, and $S_{i,f} = 3$ denotes the task is offloaded and computed by the cloud.

3.2 Time Consumption Mode

The time for a single user consumed consists of the waiting time, executing time, and transmitting time. The model of the waiting time in the cloudlet is imitated by the Fig. 2. When the tasks are chosen to be offloaded to the cloudlet, which obeys the rule that first comes will be the first severed. As all tasks arrive randomly and the computing resources in cloudlets are limited, queuing occurs when the number of arriving tasks more than the computing number within capacity.

Average Waiting Time. It is hypothesized that the service time of the cloudlet is subjected to the parameter μ, which presents the negative exponential distribution. And the interval of the time that task arrived at cloudlet from the mobile devices follows the parameter λ, which presents the negative exponential distribution. F indicates the number of virtual machines distributed in the cloudlet, which indicates the computing power of the cloudlet. The waiting time mode is established according to the basis of the queuing theory. The possibility of inactive cloudlet is shown as:

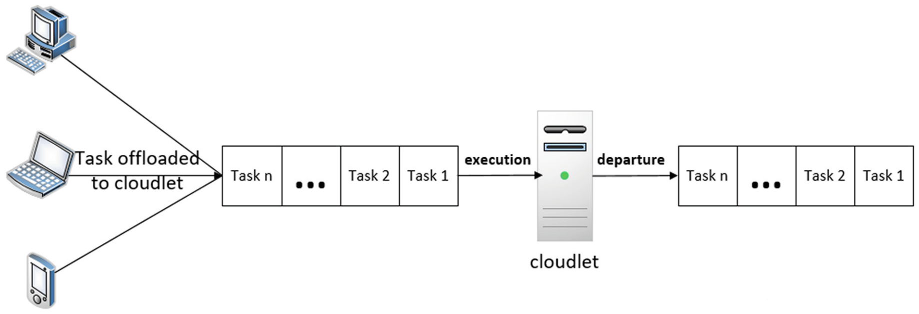


Fig. 2. The task queue in cloudlet

$$P_{idle} = [\sum_{n=0}^{F-1} \frac{\rho^{n_f}}{n_f!} + \frac{\rho^F}{F!(1-\rho_F)}]^{-1} \tag{1}$$

where $\rho = \frac{\lambda}{\mu}$ denotes the cloudlets utilization and $\rho_F = \frac{\rho}{F}$. $n_f$represent the f_{th} workflow's queue length. When the cloudlet is working normally, p_{n_f} shows the possibility of the queue size getting to n_f. Then, p_{n_f} is given as:

$$p_{n_f} = \begin{cases} \frac{\rho^{n_f}}{F!F^{n_f-F}} \cdot p_{idle} & n_f \geq F \\ \frac{p^{n_f}}{n_f!} \cdot p_{idle} & n_f < F \end{cases} \tag{2}$$

When $n_f \geq F$, we use C_w to indicate the possibility of the waiting tasks in cloudlet. And it is denoted as

$$C_w(F, \rho) = \sum_{n_f=F}^{\infty} p_{n_f} = \frac{\rho^F}{F!(1-\rho_F)} \cdot p_{idle} \tag{3}$$

Based on the theory, L_q represents the average waiting length of the queue and L_F represents the waiting length of the current queue and is calculated by

$$L_q = \sum_{n_f=F+1}^{\infty} (n_f - F)p_{n_f} = \frac{p_{idle} \cdot \rho^F}{F!} \cdot \sum_{n_f=F}^{\infty} (n_f - F)\rho_F^{n_f-F} \tag{4}$$

$$L_F = L_q + \rho \tag{5}$$

Generally speaking, the average time of tasks waiting in the cloudlet is given as

$$T_{wait} = \frac{L_F}{\lambda} - \frac{1}{\mu} = \frac{1}{F \cdot \mu - \lambda} C_w(F, \rho) \tag{6}$$

Processing Time and Transmission Time
According to the waiting time model, the execution time of tasks with different strategies $s_{i,f}$, which offload tasks to different locations is exactly computed. w_f represents the f_{th} WA's workload. f_l, f_{cl} and f_c show the computation capacity of the MD, cloudlet and cloud. The latency time of WAN and LAN is denoted as L_{WAN}, L_{LAN}. The execution time model is shown below.

$$T_{exe}^{s_{i,f}}(v_{i,f}) = \begin{cases} \dfrac{w_f}{f_l} & s_{i,f}=1 \\ T_{wait} + \dfrac{w_f}{f_{cl}} + L_{LAN} & s_{i,f}=2 \\ \dfrac{w_f}{f_c} + L_{WAN} & s_{i,f}=3 \end{cases} \tag{7}$$

The time spent during the data transmission among MD, cloud, and cloudlet is computed by

$$T_{tran}^{(s_{i,f},s_{j,f})}(v_{i,f},v_{j,f}) = \begin{cases} 0 & S_t=1 \\ \dfrac{d_{i,f}}{B_{cl}} & S_t=2 \\ \dfrac{d_{i,f}}{B_c} & S_t=3 \end{cases} \tag{8}$$

If the tasks $v_{i,f}$ and $v_{j,f}$ are executed in the same locations or they are executed in the cloudlet and the cloud respectively, for instance, $(s_{i,f}, s_{j,f}) \in \{(2,3),(3,2),(1,1),(2,2),(3,3)\}$, the transmission time will be equal to 0, and we define this situation as $S_t = 1$. In addition, when $v_{i,f}$ and $v_{j,f}$ are executed on the different locations, such as $(s_{i,f}, s_{j,f}) \in \{(1,2),(2,1)\}$, the task data is transmitted via the network LAN, where its bandwidth is depicted as B_{cl} and $S_t = 2$. Furthermore, when $(s_{i,f}, s_{j,f}) \in \{(1,3),(3,1)\}$, the task data is transmitted via the network WAN, where its bandwidth is B_c, and $S_t = 3$. The total time consumption is given by

$$T_{total}(S_{i,f}) = \sum_{i=1}^{N_f} T_{exe}^{s_{i,f}}(v_{i,f}) + \sum_{r(v_{i,f},v_{j,f}\in\xi)} T_{tran}^{(s_{i,f},s_{j,f})}(v_{i,f},v_{j,f}) \tag{9}$$

3.3 Energy Consumption Model

The energy consumed for processing and transmission makes up the total energy consumption. The processing energy consumption of the task $v_{i,f}$ is represented by $E_{exe}^{s_{i,f}}(v_{i,f})$ and $E_{trans}(v_{i,f}, v_{j,f})$ denotes the energy consumption which is caused by the data transmission between the task $v_{i,f}$ and the task $v_{j,f}$. The formulation is given as

$$E_{exe}^{s_{i,f}}(v_{i,f}) = \begin{cases} \dfrac{w_{i,f}}{f_l}\cdot p_A & s_{i,f}=1 \\ (T_{wait} + \dfrac{w_{i,f}}{f_{cl}} + L_{LAN})\cdot p_I & s_{i,f}=2 \\ (\dfrac{w_{i,f}}{f_c} + L_{LAN})\cdot p_I & s_{i,f}=3 \end{cases} \tag{10}$$

where the consumed power when MD is in the active state is denoted as p_A and p_I denotes the consumed power of inactive MD. The energy consumption caused by transmission between task $v_{i,f}$ and task $v_{j,f}$ is shown as

$$E_{tran}^{(s_{i,f},s_{j,f})}(v_{i,f}, v_{j,f}) = \frac{d_{i,j}}{B} \cdot p_T \tag{11}$$

where p_T shows the transmitted power of the MD. Thus, the total energy consumption of the f_{th} workflow is given as

$$E_{total}(S_{i,f}) = \sum_{i=1}^{N} E_{exe}^{s_{i,f}}(v_{i,f}) + \sum_{r(v_{i,f},v_{j,f}\in\xi)} E_{tran}^{(s_{i,f},s_{j,f})}(v_{i,f}, v_{j,f}) \tag{12}$$

3.4 Problem Formulation

The objective of the offloading issue is not only to minimize the total consumption of energy, but also optimize the execution time of the WAs while meeting timing constraints of WAs. The objective function is calculated by

$$Min\ E_{total}(S_{i,f}), \forall f \in \{1, 2...F\} \tag{13}$$

$$Min\ T_{total}(S_{i,f}), \forall f \in \{1, 2...F\} \tag{14}$$

$$s.t. T_{total}(S_{i,f}) \le T_{ddl}, \forall f \in \{1, 2...F\} \tag{15}$$

$$s_i \in \{1, 2, 3\} \tag{16}$$

where T_{ddl} shows the constrained time of the f_{th} WA.

4 Multi-objective Offloading Algorithm for Hybrid Workflow Applications

In this section, the details of our algorithm are illustrated. Firstly, in order to make PSO algorithm more suitable for solving discrete problem, we redefine some basic parameters. Then, the basic steps of multi-objective offloading algorithm for hybrid WAs(MOHWA) are described. Meanwhile, the structure and the construction method of hybrid WAs are depicted.

4.1 Preliminary

PSO is an effective and efficient algorithm inspired by the foraging behavior of animals where each particle in the group can update its velocity and direction dynamically through continuous learning of the group and its activities so as to improve the collective interests effectively. Originally, PSO algorithm was proposed for solving continuous space problems in 1995 [16]. Later, in 1997, the discrete binary version of the algorithm was presented to operate on discrete binary variables [17].

PSO propose an efficient global search algorithm based on the group intelligence. Firstly, PSO distributes the particles into solution space randomly. Each particle has a position, velocity and other information, and we will record the optimal position of the particle. Then, a fitness function is designed to evaluate the position of particles. Before each movement of particles, it will search the nearby area first, and compare the position of the nearest particle to the target with the best position in its own memory and adjusts its speed and direction according to these, until the particles in the group pass through the target position or reach the upper limit of iteration times.

PSO is often used to solve the continuous problems, but in solving discrete problems, the performance of PSO is not very well. Therefore, we redefine some parameters and attributes in PSO to solve discrete offloading strategy problems. The redefinition is as follows:

Definition 1. The Properties of Particles: As shown in Fig. 3, each task is an individual in the algorithm, also known as a particle, and each WA is a group, also known as population. Every individual in the swarm can search for the best position through his own experience and the experience of the group by vectors adding.

Definition 2. The offloading paths of particles: Defining the location of each particle is its offloading strategy. Each WA consists of many tasks. When tasks are offloaded, they have three paths to choose. We define the offloading path $d \in \{1, 2, 3\}$, and the position of particles $P_i^k(d, n) \in \{0, 1\}$. If $P_i^k(1, n) = 1$, the task will be executed locally. If $P_i^k(2, n) = 1$, the task will be offloaded to cloudlet for execution. Correspondingly, if $P_i^k(3, n) = 1$, the task will be offloaded to the cloud.

Definition 3. The velocity of particles: We define the velocity of each particle as a matrix of $d \times 1$. Correspondingly, the velocity equation $V_i^{k+1}(d, n)$ in discrete case is constructed. Furthermore, the velocity of each WA is a matrix of $d \times n$. In this way, we can easily find out which path the particle offloaded with the fastest speed.

Definition 4. Redefining Decision Parameters: We redefine the determinant parameters of the method. Convert the velocity inertia parameter ω, the impact of the optimal location currently found by individuals c_1 and the impact of the optimal location currently found by groups c_2 from fixed to variable ones. They will change with the need of our offloading strategy. Moreover, the convergence factor α is added to prevent the potential problem of local convergence of the algorithm.

4.2 The Basic Steps of MOHWA

Compared to other algorithms, PSO has a better computing ability, which can achieve convergence in a short time, especially in solving big data issues. However, the diversity of the population in the search space may be lost, which may

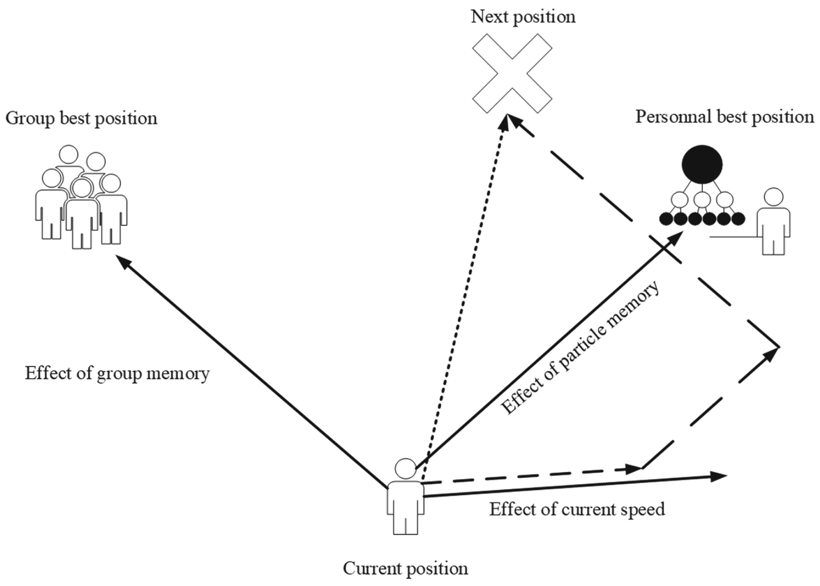


Fig. 3. The graphical representation of PSO

lead to local convergence. In order to solve this problem, we propose an improved optimization method based on PSO algorithm, named MOHWA. Our objective is to find the best offloading strategy to optimize both energy consumption and the time consumption of MDs jointly. In our algorithm, each particle is evaluated by the fitness function and the computation of the fitness value is shown as Eq. (17):

$$Fit(V_{i,j}) = E_{total}(S_i, f) + T_{total}(S_{i,f}), \forall f \in \{1, 2...F\} \tag{17}$$

Equation (17) means the smaller of the fitness, the result is better. The properties of the particle are defined by a $d \times N$ position matrix $P(d,n)$ and a $1 \times N$ velocity matrix $V(d,n)$. The position and the velocity of particles update as the algorithm iterates to find the best position, $P(d,n)$ and $V(d,n)$ is updated by using Eq. (18) and (19):

$$P_i^{k+1}(d,n) = \begin{cases} 1 & if(V_i^{k+1}(d,n) = maxV_i^{k+1}(d,n)) \\ 0 & otherwise \end{cases} \tag{18}$$

$$\begin{aligned} V_i^{k+1}(d,n) = {} & wV_i^k(d,n) \\ & + c_1 r_1((pbest_i(d,n)) - X_i^k(d,n)) \\ & + c_2 r_2((gbest_i(d,n)) - X_i^k(d,n)) \end{aligned} \tag{19}$$

In Eqs. (18) and (19), V_i^k and P_i^k refer to the velocity and position of the task in the k-th iteration. In Eq. (18), the position of particles is updated by comparing their velocities. In Eq. (19), w is inertia weight, which is used to measure the effect of current individual speed, the larger w, the greater the impact of current individual speed vice versa. Similarly, c_1 and c_2 are used to measure the impact of the optimal location currently found by individuals and groups to the next offloading respectively. The greater the values of c_1 and c_2, the greater the impact of the corresponding factors. Finally, the position of the current individual can be updated by adding vectors.

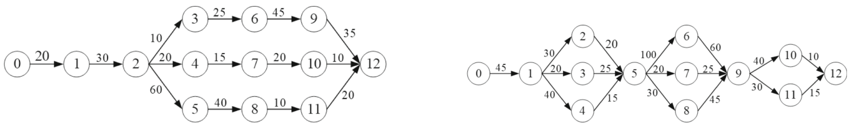
Algorithm 1. PSO-based location updating method

```
Input: Attributes of Workflow and Deadline $W_f = (V, \xi)$, $T_{ddl}$;
Output: Optimal location $S^* = (s_1^*, s_2^*, \cdots, s_n^*)$, Time, Energy;
 1: Predefine the bandwidth of LAN $B_c$ and the bandwidth of WAN $B_{cl}$
 2: Predefine the computation capacity $f_c$ and $f_{cl}$
 3: Set the cloudlet parameters $\lambda, \eta$, and $MV$
 4: Initialize the key parameters $w, c_1, c_2, r_1, r_2$
 5: Initialize the position of particles $P_i = p_1, p_2, \cdots, p_n$, $p_n \in \{1, 2, 3\}$
 6: Set the convergence factor $\alpha \in (0, 1)$
 7: $gbest_{new} \leftarrow 0, D_p^i \leftarrow 0, D_g^i \leftarrow 0, pbest_{new}^i \leftarrow 0$
 8: for each $i \in \theta$ do
 9:     Use (9) and (12) calculate the extremum of two targets $pbest_e^i$, $pbest_t^i$, $gbest_e$
    and $gbest_t$
10: end for
11: while ($k \ll I_{max}$) or ($gbest_{new} \ll I_{stop}$) do
12:     $gbest_{new} = Average(gbest_e, gbest_t)$
13:     for each $i \in \theta$ do
14:         $D_p^i = Abs(pbest_e^i - pbest_t^i)$
15:         $D_g = Abs(gbest_e - gbest_t)$
16:         if $D_p^i < D_g$ then
17:             $pbest_{new}^i = RandSelect\{pbest_e^i, pbest_t^i\}$
18:         else if $D_p^i \gg D_g$ then
19:             $pbest_{new}^i = Average\{pbest_e^i, pbest_t^i\}$
20:         end if
21:         Bring $pbest_{new}$ and $gbest_{new}$ into (18) and (19) to renewal $P$ and $V$
22:     end for
23:     for each $i \in \theta$ do
24:         if $\alpha = 1$ then
25:             $p_i = RandSelect\{1, 2, 3\}$
26:         end if
27:     end for
28: end while
```

Algorithms 1 describes the basic steps of MOHWA. The input is attribute $W_f = (V, \xi)$ of WAs and the constrained deadline T_{ddl}, the output is the optimal offloading strategy. Firstly, we set B_c, B_{cl}, f_c f_{cl} and some key parameters used in calculation (line 1–4). Next, we need to randomize the initial position $P_i = (p_1, p_2, \cdots, p_n)$. In this experiment, the offloading path is set to three levels: local execution, offloading to cloudlet execution and offloading to cloud execution, which is expressed as $p_n \in \{1, 2, 3\}$ (line 5). Then we define the scope of α, which is the convergence factor and effectively prevents local convergence and greatly improves the inherent shortcomings of PSO (line 6). Some other characters that need to be predefined are to prepare for iteration (line 7). Next, updating the extremum of the particles (line 8–10). $gbest_{new}$ is a trade-off between the optimal values of a population on two objectives, in which $Average$ can be an average or a dynamic selection of a certain proportion (line 11–12). D_p^i and D_g are predefined quantities for the next two-objective optimal solution, and their values change dynamically with the change of iteration (line 13–15). Then, according to the comparison of D_p^i and D_g, the value of $pbest_{new}^i$ can be obtained (line 16–20). Using $gbest_{new}$ and $pbest_{new}^i$ instead of the original $gbest$ and $pbest$ in Eq. (19), which calculate V and P are the result of multi-objective considerations (line 21). Finally, the convergence factor α is used to further modify the results to prevent local convergence of the results (line 22–28).

4.3 The Mechanisms of Hybrid Workflow Applications

When solving hybrid WAs synthetically based on PSO with two objectives of energy consumption and time consumption. The emphasis is on the establishment of WAs model and how to avoid the disadvantage that PSO is easy to converge locally. In order to optimize hybrid WAs, we define a set of ordered WAs and a set of unordered WAs.



(a) graph1 (b) graph2

Fig. 4. The collection of sequential WAs

As shown in Fig. 4, there are two ordered WAs, each WA has 13 subtasks and each subtask represents a particle in the swarm. There is a certain amount of assignments that need to be transferred between two tasks. Each task may have several precursor tasks and several successor tasks. For each task, it can only be performed if all of its predecessor tasks are finished. For example, for task 5 in Fig. 4.(b), it can only be computed when task 2, task 3 and task 4 have been finished.

Similarly, as shown in Fig. 5, a set of unordered WAs are defined, and the relationship between tasks is more stochastic than the two WAs in Fig. 5. In this way, we can construct three different input situations: two ordered WAs, two ordered WAs and an unordered WA, two ordered WAs, as well as two unordered WAs. Moreover, comparing energy consumption and time consumption of MDs can also test the adaptability of our method efficiency.

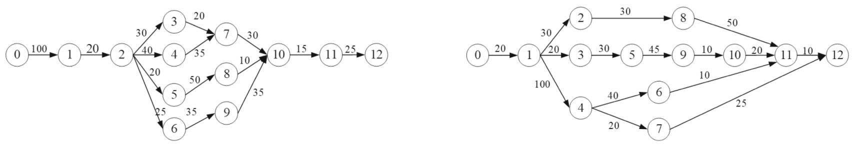

(a) graph3 (b) graph4

Fig. 5. The collection of unordered WAs

5 Experimental Evaluation

In this section, a series of comparative experiments are conducted to evaluate the performance of MOHWA. First, in order to test efficient, we preset a series of relevant parameters. Then, we evaluate the performance of MOHWA by comparing it with the other three traditional task offloading paths, and on this basis, we increase the number of WA to evaluate the performance further.

5.1 Experimental Settings

In order to analyze the effectiveness of the proposed method, we construct some other task offloading methods besides MOHWA. The specific methods are as follows.

No Offloading(NO): All tasks are completed on MDs, which is also the traditional task execution scheme, defined as NO. There is no transmission energy consumption in the execution of tasks, but it may cause a heavy hardware burden on local devices as the limitation of local computing efficiency and resources.

Offloading to Cloudlet Completely (OCCL): All tasks are offloaded to the cloudlet for calculation, defined as OCCL. This is a more efficient strategy, which allows faster computation, but the computing resources on the cloudlet are limited. When the amount of tasks is very large, the task is not suitable for execution on the cloudlet.

Offloading to Cloud Completely (OCC): All tasks are offloaded to could for calculation, defined as OCC. This strategy can reduce the consumption of computing greatly, which has almost infinite computing resources. However, there will be a large transmission delay in task transmission compared with other offloading methods.

MOHWA: One part of tasks are offloaded to the cloudlet, the other part are offloaded to cloud, and the rest of them are completed on MD.

In our experiment, the methods are implemented base on MATLAB language on a physical machine with 2 Intel Core i5-5200U 2.20 GHz processors and 4 GB RAM and the operating system is Win7 64. Specific settings are shown in Table 1.

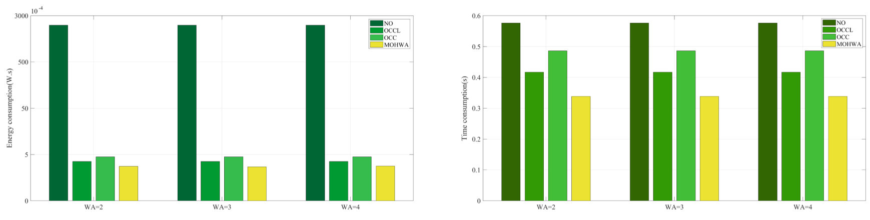
Table 1. Simulation environment

Parameter	Value
MDs' power when CPU is idle	0.001 W
MDs' power when CPU is busy	0.5 W
Task transfer capabilities of MDs	0.1 W
Task processing capabilities of MDs	500 MHZ
The computing competence of the cloudlet	2000 MHZ
The computing competence of the cloud	3000 MHZ
The latency of LAN	1 ms
The latency of WAN	30 ms
The bandwidth of LAN	100 kb/s
The bandwidth of WAN	50 kb/s

5.2 Performance Evaluation

In this paper, both energy consumption and the time consumption of each WA under the constraint of time are considered, and the benefits are assessed by fitness function. As shown in Fig. 4 and Fig. 5, four WAs are used for experimental evaluation, each WA contains 13 tasks. There are certain constraints and natural attributes between each service. The quantity for each service is set as $\omega = \{5, 14, 20, 5, 16, 13, 68, 55, 23, 53, 10, 22, 5\}$, and the dependence is $\varepsilon_i \in \{1, 2, 3\}$. The relationship between the tasks determines the transmission consumption between them. There are six situations. When $\varepsilon_i = \varepsilon_j$, both tasks are completed at the same location, the transmission consumption $E_{tran} = 0$. When $\varepsilon_i = 1, \varepsilon_j = 2$, we need to use Eq. (12) to calculate the consumption. Similarly, when $\varepsilon_i = 1, \varepsilon_j = 3$ or $\varepsilon_i = 2, \varepsilon_j = 3$ we need to use the corresponding equation to solve its consumption.

As shown in Fig. 6, MOHWA is compared with the other three methods. When WA = 2, the input is a mixture of two ordered WAs. When WA = 3, the input is a mixture of two ordered WAs and one unordered WA. When WA = 4, the input is a mixture of two ordered WAs and two unordered WAs. These

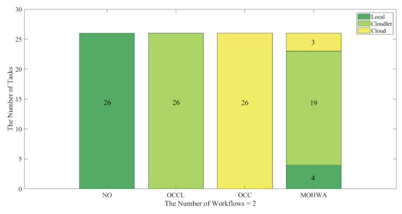


(a) Energy consumption (b) Time consumption

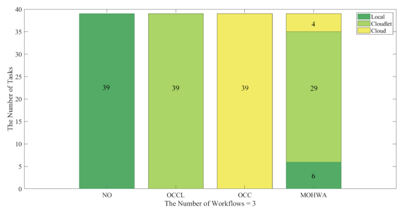
Fig. 6. Comparison of two objectives with different strategies

three scenarios have included almost all possible scenarios in hybrid WA, greatly simplifying the experimental process. As the number of WAs increases, the consumption of each method also rises, but we can see that MOHWA still has good advantages, and through the analysis of the offloading results, we can get that MOHWA is still feasible although the input is multi-WA.

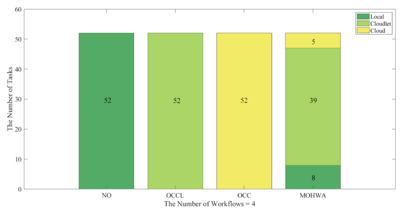
Then, in order to explore how to optimize the unloading strategy concretely, we analyze the offloading path of tasks. As shown in Fig. 7, we explore the specific offloading paths of the three hybrid WAs we contributed. We can know that most of the tasks will be offloaded to the cloudlet because cloudlet has a high computing speed and a closer distance between the mobile terminal and the cloudlet, which also has a smaller transmission cost. However, there are still a few tasks that have not been offloaded and are still computed locally. This is because it does not generate the cost of task transmission and is suitable for computing with small amount of task. Moreover, a small portion of the rest tasks will be offloaded to the cloud for computation. Although the task offloaded to the cloud will generate a lot of transmission cost, the tasks with a long queue time delay are suitable to be offloaded to cloud due to its strong computing power and nearly infinite computing resources. MOHWA achieves optimization by allocating offloading strategies reasonably.



(a)The number of WAs = 2



(b)The number of WAs = 3



(c)The number of WAs = 4

Fig. 7. The number of tasks offloaded to each path

6 Conclusion

In this paper, the multi-objective computation offloading problem for hybrid WAs in MEC environment are investigated. To tackle this problem, we have proposed a multi-objective computation offloading algorithm for hybrid WAs, which generates the optimal application strategy while meeting the constrained deadline of WAs. Extensive experimental evaluations have conducted to show the efficiency and effectiveness of our proposed method.

In the future, we will focus on the computing offloading problem under the dynamic changing circumstance of some key factors such as the queueing waiting time and the power of edge servers. AdditioAccording to the waiting timenally, we will improve the mathematic mode and the efficiency of optimization method further, such as the convergence speed. In addition, the computation offloading for WAs in multi-cloudlet scenario will be studied [21,22].

Acknowledgements. This work is supported by the National Science Foundation of China (Grant No. 61902133), the Natural Science Foundation of Fujian Province (Grant No. 2018 J05106), the Education and Scientific Research Projects of Young and Middle-aged Teachers in Fujian Province (JZ160084), and the Scientific Research Foundation of Huaqiao University under Grant No. 14BS316, Quanzhou Science and Technology Project (No. 2015Z115), the Fundamental Research Funds for the Central Universities (No. 30918014108).

References

1. Qi, L., et al.: Finding all you need: web APIs recommendation in web of things through keywords search. IEEE Trans. Comput. Soc. Syst. (2019)
2. Xu, X., Liu, Q., Zhang, X., Zhang, J., Qi, L., Dou, W.: A blockchain-powered crowdsourcing method with privacy preservation in mobile environment. IEEE Trans. Comput. Soc. Syst. (2019). https://doi.org/10.1109/TCSS.2019.2909137
3. Zhang, Y., et al.: Covering-based web service quality prediction via neighborhood-aware matrix factorization. IEEE Trans. Serv. Comput. (2019). https://doi.org/10.1109/TSC.2019.2891517
4. Qi, L., Chen, Y., Yuan, Y., Fu, S., Zhang, X., Xu, X.: A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. World Wide Web J., 1–23 (2019)
5. Shi, W., Cao, J., Zhang, Q., Li, Y., Xu, L.: Edge computing: vision and challenges. IEEE Internet Things J. **3**(5), 637–646 (2016)
6. Zhang, W., Guo, B., Shen, Y., Wang, Y., Xiong, W., Duan, L.: Computation offloading on intelligent mobile terminal. Chinese J. Comput. **39**(5), 1021–1038 (2016)
7. Mach, P., Becvar, Z.: Mobile edge computing: a survey on architecture and computation offloading. IEEE Commun. Surv. Tutorials **19**(3), 1628–1656 (2017)
8. Peng, K., Leung, V., Xu, X., Zheng, L., Wang, J., Huang, Q.: A survey on mobile edge computing: focusing on service adoption and provision. Wireless Commun. Mob. Comput. **2018**, 16 (2018). https://doi.org/10.1155/2018/8267838. Article ID 8267838
9. Jia, M., Cao, J., Yang, L.: Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing. In: Proceedings of the IEEE Conference on Computer Communications Workshops, INFOCOM Workshops 2014, pp. 352–357 (2014). https://doi.org/10.1109/INFCOMW.2014.6849257

10. Deng, S., Huang, L., Taheri, J., Zomaya, A.Y.: Computation offloading for service workflow in mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. **26**(12), 3317–3329 (2015). https://doi.org/10.1109/TPDS.2014.2381640
11. Zhou, Y., Li., Z., Ge, J., Li, C., Zhou, X., Luo, B.: Multi-objective workflow scheduling algorithm based on delay transmission mechanism in mobile cloud computing environment. J. Softw. **29**(11), 1–20 (2018)
12. Wu, H., Knottenbelt, W., Wolter, K., Sun, Y.: An optimal offloading partitioning algorithm in mobile cloud computing. In: Agha, G., Van Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 311–328. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43425-4_21
13. Xu, X., Dou, W., Zhang, X., Chen, J.: EnReal: an energy-aware resource allocation method for scientific workflow executions in cloud environment. IEEE Trans. Cloud Comput. **4**(2), 166–179 (2015)
14. Wang, S., Zhao, Y., Xu, J., Yuan, J., Hsu, C.: Edge server placement in mobile edge computing. J. Parallel Distrib. Comput., 160–168 (2019)
15. Satyanarayanan, M., Lewis, G., Morris, E., Simanta, S., Boleng, J., Ha, K.: The role of cloudlets in hostile environments. IEEE Pervasive Comput. **12**(4), 40–49 (2013)
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: International Conference on Neural Networks, vol. 4, pp. 1942–1948 (1995)
17. Kennedy, J., Eberhart, R.: A discrete binary version of the particle swarm algorithm, pp. 4104–4108 (1997)
18. Jia, M., Liang, W.: Delay-sensitive multiplayer augmented reality game planning in mobile edge computing. In: Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, pp. 147–154. ACM (2018)
19. Li, B., He, M., Wu, W., Sangaiah, A.K., Jeon, G.: Computation offloading algorithm for arbitrarily divisible applications in mobile edge computing environments: an OCR case. Sustainability **10**(17), 196–210 (2018)
20. Zhang, J., et al.: Hybrid computation offloading for smart home automation in mobile cloud computing. Pers. Ubiquit. Comput. **22**(1), 121–134 (2018)
21. Roy, D., De, D., Mukherjee, A., Buyya, R.: Application-aware cloudlet selection for computation offloading in multi-cloudlet environment. J. Supercomput. **73**(4), 1672–1690 (2017)
22. Liu, L., Fan, Q.: Resource allocation optimization based on mixed integer linear programming in the multi-cloudlet environment. IEEE Access, 1 (2018)