IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2018, accepted September 13, 2018, date of publication September 18, 2018, date of current version October 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2870915

A Deadline-Constrained Multi-Objective Task Scheduling Algorithm in Mobile Cloud Environments

LI LIU[1], QI FAN[1], AND RAJKUMAR BUYYA[2], (Fellow, IEEE)
[1]School of Automation and Electrical Engineering, University of Science and Technology Beijing, Beijing 100083, China
[2]Cloud Computing and Distributed Systems Laboratory, School of Computing and Information Systems, The University of Melbourne, Parkville Campus, Melbourne, VIC 3010, Australia

Corresponding author: Qi Fan (s20160587@xs.ustb.edu.cn)

This work was supported by the National Natural Science Foundation of China under Grants 61772068, 61370132, and 61472033.

ABSTRACT By leveraging the technology of the mobile cloud computing, resource capacity, and computing capability of mobile devices could be extended. However, it is difficult to schedule tasks submitted by mobile users when the number of tasks and service providers increases and to optimize multiple objectives while satisfying users' requirements. In this paper, the task scheduling is modeled as a multi-objective optimization problem, and we consider both unconstrained and time deadline constrained cases. To address this problem, a heterogeneous earliest finish time (HEFT) using technique for order preference by similarity to an ideal solution method is proposed, which is named as HEFT-T algorithm. For the unconstrained case, a three-stage strategy based on HEFT-T algorithm is presented to select the optimal solutions by applying non-dominated sorting approach. For the deadline-constrained case, an adaptive weight adjustment strategy based on HEFT-T is proposed to adjust weight value for time. Compared with other of the state-of-the-art algorithms, our proposed algorithm performs better in the criterion of both the optimization for total cost as well as mean load, and the deadline-constraint meeting rate.

INDEX TERMS Mobile cloud computing, task scheduling, deadline-constrained, multi-objective optimization.

I. INTRODUCTION

Mobile cloud computing (MCC) is the integration of mobile computing and cloud computing, which aims to overcome the disadvantages of limited resource compacity and computational capability by migrating the complicated computing-intensive tasks from the mobile device to the cloud [1], [2]. It is convenient for mobile users to obtain computing resources according to their demands.

In the MCC environment, the task set is composed of a series of tasks, which could be scheduled to the proper resource nodes according to the requirements of various mobile users. The resource nodes here mean different kinds of service providers. By taking computing capability, communication delay and other performance factors of each node into consideration, tasks could be scheduled appropriately in terms of various demands. For example, tasks can be offloaded to the cloud data center (such as MAUI and CloneCloud) [3], cloudlet [4] or Virtual Machines (VMs) in order to extend the computing capability of mobile devices. However, when the number of tasks and service providers increases, computing cost rises as well. In addition, communication delay, which caused by the various distribution of cloud resources, would be different if tasks are scheduled to various clouds. Thus, it is quite significant to consider how to optimize the task scheduling while satisfying the different demands of users.

A number of works have investigated the aforementioned issues. The Heterogeneous Earliest Finish Time (HEFT) [5] is a popular task scheduling algorithm which considers task priority and processor selection, while it mainly focuses on single-objective and does not consider about constraints. A Cooperative Multi-tasks Scheduling based on Ant Colony Optimization algorithm (CMSACO) was proposed to achieve optimum profit with constraints of finish time and resource capacity [6]. However, in the real life, it is more reasonable to consider the task scheduling problem under MCC environment as a multi-objective problem with constraints because of the various optimization objectives and different

requirements from mobile users. By taking this issue into consideration, the Enriched-Look ahead HEFT algorithm (E-LHEFT) [7] was proposed to solve the task scheduling problem in Mobile Cloud environment while optimizing both Quality of Service (QoS) and load balancing without considering constraints. A multi-objective simulated annealing algorithm combined with Pareto theory was represented to solve a triple objective optimization problem of energy, reliability and Quality of Experience (QoE) while meeting the constraints of load, total time deadline and energy of mobile devices [8]. However, it did not highlight how to process these constraints.

Therefore, this paper focuses on how to reasonably allocate the tasks to resource nodes to minimize the total cost and load. Two cases are discussed in this paper, which are unconstrained and constrained with total execution time respectively. In these two cases, the time constraint is addressed as an optimization objective. In the first case, a HEFT-T algorithm is proposed, which combines HEFT algorithm with TOPSIS method to solve the multi-objective problem. TOPSIS is to select an alternative which is closest to the ideal solution and farthest from the negative-ideal solution [9]. As different weight values of each objectives in TOPSIS result in various solutions, a set of solutions is obtained by setting the weight value of each objective from 0 to 1. In order to obtain the optimal solution, a three-stage strategy is proposed. Firstly, a set of solutions is calculated under different weight values. Then, Pareto solutions are selected by utilizing the Non-dominated Sorting [10]. Finally, the optimal one closest to the ideal solution is chosen among Pareto solutions. In the second case, based on our proposed algorithm, an adaptive weight adjustment strategy is proposed. The optimal weight value obtained in the first case is utilized, and the weight value of time objective can be adjusted adaptively in order to meet the time deadline.

To summarize, the major contributions of this paper are as follows.

- The task scheduling under MCC environments is conducted as a multi-objective optimization problem, which takes both unconstrained and constrained with time deadline cases into consideration.
- A HEFT-T algorithm is proposed for task scheduling to minimize both total cost and mean load under MCC environment.
- In the unconstrained case, a three-stage strategy is presented, which aims at selecting the optimal solutions in terms of Non-dominated Sorting.
- In the constrained case, an adaptive weight adjustment strategy based on the proposed algorithm is proposed to adjust weight values adaptively to satisfy the deadline. This approach is verified under 4 deadlines.
- Simulations show that our proposed algorithm has better performance in terms of total cost, the mean load and the deadline meeting rate compared with other existing task scheduling algorithms.

The rest of the paper is organized as follows. Section 2 briefly reviews the related works of algorithms and approaches of task scheduling. Section 3 describes the system and task scheduling models. In section 4, the unconstrained and deadline-constrained algorithms based on HEFT-T are presented in detail. Section 5 compares the experimental results with other of the state-of-art algorithm and section 6 concludes the paper.

II. RELATED WORKS

For the task scheduling problem under MCC environment, the tasks could be represented independently [21] or as task flow graphs [6] (e.g. Directed Acyclic Graph (DAG) [25]). Like in the cloud environment, some optimization objectives are crucial for users' and providers' expectations, such as energy consumption, QoS [26], [27], cost [28] and so on. Thus, we will review the state-of-the-art works by considering their optimization objectives.

A. TASK SCHEDULING WITH ENERGY CONSUMPTION OPTIMIZATION

Li *et al.*, [14] proposed an Energy-aware Dynamic Task Scheduling (EDTS) algorithm in the MCC environment. This algorithm aims to minimize the total energy consumption of smartphones based on Dynamic voltage scaling (DVS) technique while satisfying the strict time deadline and the probability constraint for applications, where the probability constraint for applications here means the probability of tasks executed successfully. A task scheduling algorithm was represented to solve the energy consumption problem of mobile devices under Mobile Cloud environment [15]. There are mainly three steps in this algorithm. Firstly, the initial scheduling scheme was obtained by minimizing total execution time. Then, a task migration method was applied to migrate tasks to the cloud while satisfying the deadline of applications in order to reduce the power consumption of mobile devices. Finally, the Dynamic Voltage and Frequency Scaling (DVFS) technique was utilized to further decrease the energy consumed by mobile devices while meeting the time deadline. The joint scheduling and computation offloading (JSCO) concept for multi-component applications was proposed in [19], which makes the optimal decision for which components should be offloaded to cloud and the scheduling order of these offloaded components. Integer linear programming technique is utilized to maximize the saved energy of remote execution.

B. TASK SCHEDULING WITH COST OPTIMIZATION

Wu *et al.* [29] proposed two algorithms named Probabilistic List Scheduling (ProLiS) and L-ACO respectively. The main purpose of this paper is to minimize the execution cost with a user-defined deadline constraint. Hung *et al.* [18] proposed an algorithm which is the extension of Contention Aware Scheduling (CAS) algorithm. Firstly, it determines the executing order of each task by setting their priority. Then, the most proper processor was chosen to execute

the selected task according to the minimum trade-off solutions between total cost and finish time. Two problems were taken into consideration while scheduling tasks in the MCC environment [20]. The first is about which tasks should be offloaded to the cloud servers, and the other is the optimal price of cloud servers. Thus, this paper considers both task scheduling and pricing strategy of cloud service servers while offloading and scheduling these tasks. For mobile users, a utility maximization function was conducted by considering energy consumption, delay and price of cloud service. For cloud service provider, the Convexification and Primal-dual methods (CoPe) was proposed to determine the optimal pricing strategy. Hung and Huh [16] introduced an improved Genetic Algorithm (GA), which minimizes the trade-off solution between task processing time and execution cost in order to achieve higher performance.

C. TASK SCHEDULING WITH TIME OPTIMIZATION

A demand-driven task scheduling model was proposed [21], and it brings an estimate method to predict the finish time of tasks. This paper copes with tasks scheduling from the perspective of both mobile users and service providers. A 2D Chromosome Genetic Algorithm (2DCGA) based on objective weighted method was proposed to minimize the finish time for mobile users and to achieve load balancing for service providers. A stochastic computation task scheduling policy was proposed in [30]. By considering the average delay and the average power consumption at the mobile device, an efficient one-dimensional search algorithm was proposed in order to achieve a shorter average execution delay.

For task scheduling algorithms, one of the most classical heuristic algorithms is HEFT [5], which has two major phases. The first phase is to select tasks with higher upward rank by considering the mean execution and communication time between tasks and to sort them. Then, in the processor selection phase, the processor with minimum execution time is selected in terms of an insertion-based approach to minimize its earliest finish time. This algorithm only aims to achieve minimum finish time regardless of constraints like QoS. Thus, some works begin to extend this algorithm by considering constraints. Zheng and Sakellariou [11] proposed a Budget-constrained Heterogeneous Earliest Finish Time (BHEFT) algorithm to take time and budget deadline into consideration. Some rules are made in the service selection phase to obtain the affordable service with the earliest finish time. Similarly, Heterogeneous Budget Constrained Scheduling (HBCS) algorithm was represented to solve a single-objective scheduling problem, where processing time is optimized, and budget is set as the constraint [12]. Different from the second phase in BHEFT, the HBCS considers the trade-off of both budget and finish time for all processors to evaluate which one should be selected. Although the BHEFT and the HBCS resolve the constraints problem by extending HEFT algorithm, but the authors only concentrate on single-objective optimization.

Some studies have extended the HEFT algorithm to achieve multi-objective optimization. Durillo and Prodan [13] introduced a scheduling method named Multi-Objective Heterogeneous Earliest Finish Time (MOHEFT), which applied Pareto method to provide users a set of trade-off optimal solutions. Also, the dominance relationships and crowding distance are used to guarantee diversity while optimizing both makespan and cost.

D. TASK SCHEDULING WITH OTHER OBJECTIVES

Meanwhile, a number of works concentrate on other optimization objectives, such as QoS, load and so on.

Enriched-Look ahead HEFT algorithm (E-LHEFT) [7] was proposed to improve the processor selection phase. The Pareto principle was conducted to achieve load balancing and QoS when assigning tasks in the MCC environment. The Hybrid Ant Colony algorithm based Application Scheduling (HACAS) algorithm was proposed to address application scheduling problem based on a Hybrid Local Mobile Cloud Model (HLMCM) [17], which aims at maximizing the total profit of HLMCM while meeting the constraints for resource capacity of each service provider.

From the aforementioned related studies, we can see that most works focus on single-objective optimization with or without constraints. Although reference [7] and [13] considered task scheduling as a multi-objective problem, these two multi-objective algorithms are not able to solve the constrained problem, and there are few studies concentrating on constraints while optimizing multi-objective problem for task scheduling under MCC environment, especially those who extended HEFT algorithm. Therefore, in this paper, we focus on extending HEFT algorithm to solve multi-objective problem as well as addressing its constraints effectively.

III. SYSTEM MODEL AND PROBLEM FORMULATION

The architecture of task scheduling in the MCC environment is depicted as Fig. 1, which mainly consists of a mobile platform (MP), task scheduling center (TSC) and a cloud platform (CP) [8]. As we can see, MP is composed of mobile devices; TSC is responsible to decide which provider would these scheduled tasks execute; CP is to provide various providers for mobile users to execute tasks.

Here is the flow of the task scheduling process under Mobile Cloud environments. Firstly, all the tasks are generated by mobile devices in MP. Those tasks are collected in the Task Sequence Set (TSS) and scheduled by TSC. The most suitable providers of CP are allocated to the scheduled tasks according to the requirements of mobile users. In this paper, we mainly concentrate on the task scheduling algorithm in the TSC.

In this section, inspired by Wang *et al.* [6], we will present some models involved in task scheduling process, which includes task graph model, communication model, execution model and task scheduling model. In this

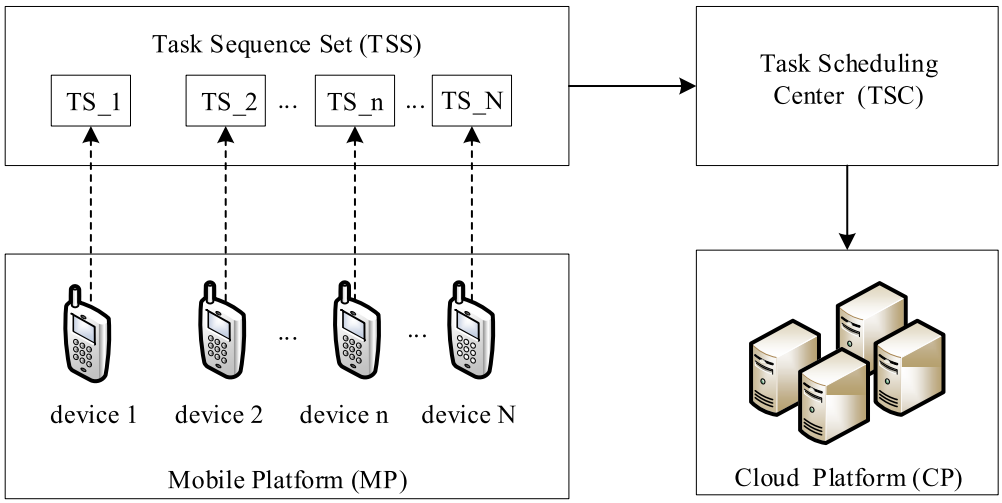

FIGURE 1. Task scheduling flow under MCC environment.

paper, task scheduling is formulated as a time-constrained, multi-objective optimization problem.

A. TASK GRAPH MODEL

A mobile application consists of multiple tasks which could be represented by a DAG. In this paper, we consider that all the tasks are executed on the cloud providers.

A task graph is conducted by the DAG, $G = (N, E)$, where N is the set of n tasks, and E is the set of $edge(i, j)$. Each $edge(i, j) \in E$ represents the communication time between task n_i and n_j, where tasks is represented as n_i, $i = 1, 2, \cdots, n$, and cloud providers are represented as p_j, $j = 1, 2, \cdots, m$.

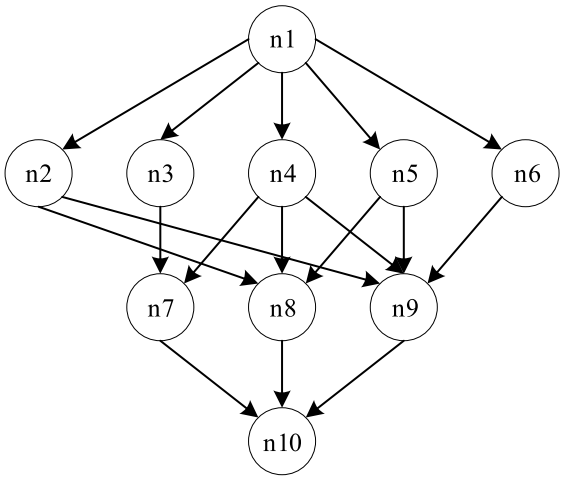

FIGURE 2. A sample of DAG.

A sample DAG is shown as Fig. 2. In the given task graph, if a task n_i does not have any predecessors, it will be considered as an entry task n_{entry}. Also, if a task n_i does not have any successors, it will be regarded as an exit task n_{exit}. We assume that task n_i cannot start until all its predecessors have finished executing, and each of the tasks only appears once in the whole scheduling process.

B. COMMUNICATION MODEL

COMMUNICATION TIME

The communication time is generated when data transfer between different providers. The amount of data that must be transferred from task n_i to task n_k can be represented as $Data = \{data_{ik} | i, k = 1, \cdots, n\}$, and the transmission rate can be represented as R. If the task n_i and task n_k are executed on different providers, for example, task n_i is scheduled on p_n and task n_k is scheduled on p_m. Then the communication time of $edge(i, k)$ is calculated as:

$$c_{ik} = data_{ik}/R \tag{1}$$

If the task n_i and task n_k are executed on the same provider, the communication time c_{ik} is set as 0.

C. EXECUTION MODEL

In the execution model, we consider three performance metrics which are completion time, cost and load respectively. Completion time is regarded as deadline constraints of mobile users; cost is the optimization objective which aims to satisfy users' expectations; and load is the optimization objective which cloud providers need to consider. By optimizing these objectives at the same time, we can fulfill both users and cloud providers expectations as much as possible. These three-performance metrics are stated in detail as following:

1) COMPLETION TIME

T_{ij} represents the execution time of task n_i when it is processed on provider p_j, which can be calculated as:

$$T_{ij} = l_i / R_j^{computing} \tag{2}$$

where $R_j^{computing}$ denotes the computing capability of provider p_j; l_i is the length of task n_i.

As introduced in [5], the attributes of Earliest Start Time (EST) and Earliest Finish Time (EFT) are applied to formulate the Total Execution Time (TET) of this task scheduling process. The earliest execution start time and earliest finish time of task n_i on provider p_j are represented as $EST(n_i, p_j)$ and $EFT(n_i, p_j)$ respectively. If it is an entry task n_{entry}, the EST can be calculated as:

$$EST(n_{entry}, p_j) = 0 \tag{3}$$

If it is one of the other tasks in the task graph, the EST and EFT are computed as:

$$EST(n_i, p_j) = \max\{M_j, \max_{n_p \in pred(n_i)}(AFT(n_p) + c_{ip})\} \tag{4}$$

$$EFT(n_i, p_j) = T_{ij} + EST(n_i, p_j) \tag{5}$$

Where $pred(n_i)$ is the task set which consists of immediate predecessor tasks of task n_i, and M_j is the accumulated execution time of tasks executed on provider p_j. After a task n_p is executed on provider p_j, the actual start time (AST) $AST(n_p)$ and actual finish time (AFT) $AFT(n_p)$ are equal to $EST(n_p, p_j)$ and $EFT(n_p, p_j)$. The EST and EFT values are calculated recursively according to Equation (3), (4) and (5) from the entry task n_{entry} to the exit task n_{exit}. Thus, the TET is defined as:

$$TET = \max\{AFT(n_{exit})\} \tag{6}$$

Here we take the situation that there is more than one exit task in the task graph into account. Thus, the TET is the maximum AFT of all the exit tasks.

2) COST

The cost of per unit time of provider p_j is represented as Cpt_j, and the cost of per unit time for transmission is denoted as Ctt. Then, if the task n_i and its predecessor task n_p are executed on the same provider, the cost of task n_i scheduled on p_j can be defined as:

$$Cost_{ij} = Cpt_j \times T_{ij} \tag{7}$$

If the task n_i and its predecessor task n_p are executed on different providers, the cost of task n_i scheduled on p_j can be defined as:

$$Cost_{ij} = Cpt_j \times T_{ij} + Ctt \times c_{ip} \tag{8}$$

3) LOAD

We assume that each provider has d dimension resources. In this paper, we only concentrate on memory and computing capability respectively. The total capacity of memory in provider p_j and computing capability of provider p_j are represented as R_j^{memory} and $R_j^{computing}$, and those two resources consumed by task n_i on provider p_j are denoted as RC_{ij}^{memory} and $RC_{ij}^{computing}$ respectively.

$$L_{ij} = \left(\frac{RC_{ij}^{memory}}{R_j^{memory}} + \frac{RC_{ij}^{computing}}{R_j^{computing}}\right)/d \tag{9}$$

D. TASK SCHEDULING MODEL

We consider minimizing the total cost and mean load of allocated providers while meeting the deadline of TET when we design the task scheduling algorithm. To facilitate the description of this task scheduling model, the assumption is listed as following:

Each task can only be executed on one provider and it cannot be partitioned. In addition, each provider can only process a task at each time.

Thus, in this paper, the task scheduling problem under MCC environment is formulated as a constrained multi-objective optimization problem.

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{m} Cost_{ij} \tag{10}$$

$$\text{Minimize: } \frac{1}{m}\sum_{i=1}^{n}\sum_{j=1}^{m} L_{ij} \tag{11}$$

$$\text{Subject to: } TET \tag{12}$$

IV. THE PROPOSED TASK SCHEDULING ALGORITHM

In this section, we consider two cases: unconstrained and constrained with TET respectively. Inspired by Ghasemi-Falavarjani *et al.* [22] and considering the low computational complexity of TOPSIS, a HEFT-T algorithm is proposed which applied TOPSIS method based on HEFT algorithm to solve those two cases respectively.

A. THE PROPOSED HEFT-T ALGORITHM

Topcuoglu *et al.* proposed HEFT algorithm [5] to achieve high performance and fast scheduling time for task scheduling. There are mainly two phases included in this algorithm, which are task selection phase and provider selection phase respectively. Therefore, our proposed algorithm based on HEFT also has these two phases.

1) TASK SELECTION

Tasks are scheduled in terms of their priorities, and tasks are given priorities and sorted according to their upward rank values. The upward rank value of a task n_i in HEFT is computed as:

$$rank_u(n_i) = \bar{T}_i + \max_{n_s \in succ(n_i)}(\bar{c}_{is} + rank_u(n_s)) \tag{13}$$

Where $succ(n_i)$ is the task set which consists of immediate successor tasks of task n_i; $\bar{T}_i$ is the average execution time of task n_i over all of the providers; and $\bar{c}_{is}$ is the average communication time of $edge(i, k)$. If a task is an exit task, its upward rank value is calculated as $rank_u(n_{exit}) = \bar{T}_{exit}$.

In HEFT algorithm, the $rank_u$ value is calculated according to the average execution and communication time, which makes the tasks with lower processing time have higher priorities. In this paper, one of the optimization objectives is cost, which is linear correlated to the processing time of scheduled tasks. Thus, similarly, in order to make tasks with lower cost obtain higher priorities, the upward rank value of

TABLE 1. Notations.

Notation	Description
n_i	The i th task
p_j	The j th provider
l_i	The length of task n_i
c_{ik}	The communication time between task n_i and task n_k
$data_{ik}$	The amount of data that must be transferred to from task n_i to task n_k
R	The transmission rate
T_{ij}	The execution time of task n_i processed on provider p_j
$EST(n_i, p_j)$	The earliest execution start time of task n_i on provider p_j
$EFT(n_i, p_j)$	The earliest finish time of task n_i on provider p_j
$pred(n_i)$	The task set which consists of immediate predecessor tasks of task n_i
$succ(n_i)$	The task set which consists of immediate successor tasks of task n_i
M_j	The accumulated execution time of tasks executed on provider p_j.
$AST(n_p)$	The actual start time of n_p
$AFT(n_p)$	The actual finish time of n_p
Cpt_j	The cost of per unit time of provider p_j
Ctt	The cost of per unit time for transmission
$Cost_{ij}$	The cost of task n_i scheduled on p_j
R_j^{memory}	The total capacity of memory in provider p_j
$R_j^{computing}$	The computing capability of provider p_j
RC_{ij}^{memory}	Memory resource consumed by task n_i on provider p_j
$RC_{ij}^{computing}$	Computing capability resource consumed by task n_i on provider p_j
L_{ij}	The mean load of task n_i on provider p_j

task n_i in proposed algorithm is defined as:

$$rank_u^c(n_i) = \overline{Cost}_i + \max_{n_s \in succ(n_i)} (\bar{c}_{is} \times Ctt + rank_u^c(n_s)) \tag{14}$$

Where $\overline{Cost}_i$ is the average execution cost of task n_i over all the providers, and Ctt is the cost of per unit time for transmission. If a task is an exit task, its upward rank value is calculated as $rank_u^c(n_{exit}) = \overline{Cost}_{exit}$.

The notations used in this paper are summarized in Table 1.

2) PROVIDER SELECTION

In HEFT algorithm, the objective is to minimize the total execution length for the task scheduling. Thus, the insertion-based policy is proposed in provider selection phase, which considers inserting a task in an earliest idle time slot between two already scheduled tasks on a processor [5]. However, in this paper, the task scheduling problem is modeled as a multi-objective optimization problem, and we have to take all the objectives into consideration. TOPSIS is a multi-criterion decision making method. It devotes to find an optimal solution which was closest from the ideal solution and longest from the negative solution simultaneously [9]. Therefore, in the provider selection phase, we apply the TOPSIS method to select the most suitable provider to optimize both total cost and mean load.

The decision matrix in this task scheduling problem contains m alternatives associated with q attributes, where x_{ij} is

TABLE 2. Pseudo code of HEFT-T algorithm.

Algorithm 1: HEFT-T algorithm
1. Initialize parameters
2. Compute $rank_u^c$ according to Equation (14) for all tasks by traversing the task graph upward, starting from the exit task;
3. Sort the tasks in a scheduling list by nonincreasing order of $rank_u^c$ values.
4. **while** there are unscheduled tasks in the list **do**
5. Select the first task n_i from the list of scheduling.
6. **for** each processor p_k **do**
7. Compute $EFT(n_i, p_k)$ value according to Equation (5)
8. Compute $Cost_{ik}$ value according to Equation (7) and (8)
9. Compute L_{ik} value according to Equation (9)
10. **end for**
11. Select the optimal provider p_j by applying TOPSIS
12. Assign task n_i to the processor p_j
13. **endwhile**
14. Return total cost, mean load and TET.

the value of jth objective when the scheduled task is executed on provider p_i.

$$D = \begin{array}{c} \\ provider\ 1 \\ provider\ 2 \\ \cdots \\ provider\ m \end{array} \begin{array}{c} \begin{array}{cccc} A_1 & A_2 & \cdots & A_q \end{array} \\ \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1q} \\ x_{21} & x_{22} & \cdots & x_{2q} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mq} \end{bmatrix} \end{array} \tag{15}$$

Firstly, the decision matrix in equation (15) is required to be normalized by following:

$$r_{ij} = x_{ij} \Big/ \sqrt{\sum_{i=1}^{m} x_{ij}^2}, \quad i = 1, 2, \cdots m,\ j = 1, 2, \cdots q. \tag{16}$$

Then, the weighted normalized decision matrix is constructed as:

$$v_{ij} = w_j \times r_{ij}, \quad i = 1, 2, \cdots m,\ j = 1, 2, \cdots q. \tag{17}$$

where w_j is the weight value of jth objective.

After calculating the weighted normalized decision matrix, the ideal (A^+) and negative- ideal (A^-) solutions can be determined as:

$$\begin{aligned} A^+ &= \{(\min_i v_{ij} | j \in J^*) \, | i = 1, 2, \cdots, m\} \\ &= \{v_1^+, v_2^+, \cdots v_j^+, \cdots v_q^+\} \end{aligned} \tag{18}$$

$$\begin{aligned} A^- &= \{(\max_i v_{ij} | j \in J^*) \, | i = 1, 2, \cdots, m\} \\ &= \{v_1^-, v_2^-, \cdots v_j^-, \cdots v_q^-\} \end{aligned} \tag{19}$$

where $J^* = \{j = 1, 2, \cdots, q | j$ associated with cost criteria$\}$ because both cost and load objectives are cost attributes.

Then, the Euclidean distance of each alternative from the ideal solution and negative-ideal one can be given by:

$$d_i^+ = \sqrt{\sum_{j=1}^{q} (v_{ij} - v_j^+)^2}, \quad i = 1, 2, \cdots m. \tag{20}$$

$$d_i^- = \sqrt{\sum_{j=1}^{q} (v_{ij} - v_j^-)^2}, \quad i = 1, 2, \cdots m. \tag{21}$$

Where d_i^+ represents the distance between the alternative and the ideal solution; d_i^- represents the distance between the alternative and the negative-ideal solution.

Therefore, the relative closeness to the ideal solution can be calculated as:

$$c_{i+} = d_i^+ / (d_i^+ + d_i^-), \quad 0 < c_{i+} < 1,\ i = 1, 2, \cdots m. \tag{22}$$

It is obvious that if c_{i+} is closer to 0, it means the alternative is closer to the ideal solution.

Finally, a set of alternatives are ranked in terms of the ascending order of c_{i+}.

Thus, the corresponding index with the minimum c_{i+} is selected as the optimal provider.

To summarize, the Pseudo code of HEFT-T algorithm is shown in Table 2.

B. UNCONSTRAINED HEFT-T ALGORITHM

As described in Section A in IV, it can be observed that different weight values of objectives can result in various solutions in provider selection. Based on the proposed HEFT-T algorithm, in order to obtain the optimal solution and the corresponding optimal weight values of each objective, we will: (1) find all the solutions when weight value of each objective changes from 0 to 1; (2) select non-dominated

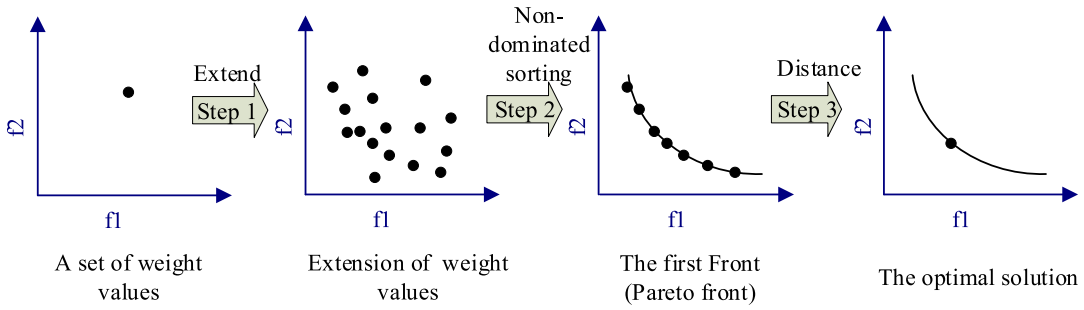

FIGURE 3. The three-stage strategy of unconstrained HEFT-T algorithm.

TABLE 3. Pseudo code of unconstrained HEFT-T algorithm.

Algorithm 2: Unconstrained HEFT-T algorithm
1. Initialize parameters
2. Obtain a solution set consisting of all the solutions calculated by Algorithm 1 where the weight value of each objective changes from 0 to 1.
3. Non-dominated sorting for the solutions in the solution set.
4. Select the first front members.
5. Calculate the ideal and negative-ideal solutions.
6. Normalize solutions in step 4.
7. Calculate the Euclidean distance of each normalized solution from ideal solution.
8. Select the one with the shortest distance as the optimal solution.
9. Select the weight values corresponding to the optimal solution as the optimal weights.

solutions from that obtained in (1) according to the Fast Non-dominated Sorting approach [10]; (3) calculate the ideal and negative-ideal solutions and find the optimal one which is the closest to the ideal solution. This three-stage strategy is illustrated in Fig. 3.

The Fast Non-dominated Sorting is an effective approach to sort all the solutions with a less computational complexity. For each solution, there are mainly two entities calculated, which are the number of solutions that dominate the solution i and the set of solutions that the solution i dominates respectively. Firstly, solutions that no one could dominate are considered as the first front. Secondly, each member in this first front is visited, and its count is reduced by one. For any member in this process, if its count becomes zero, it is put into a separate list H. After traversing all the members in the first front, a newly identified front H is generated. Then, the second step is repeated until the whole solutions are sorted. The first front that no one could dominate is considered as the Pareto front. Therefore, we use this approach to obtain the Pareto solutions.

It is worth to note that the TET is regarded as an optimization objective in this unconstrained case as the optimal weight value is utilized in the deadline-constrained HEFT-T algorithm. But this does not affect the optimization of the other two objectives, because we only care about the objectives of total cost and mean load when we select the ideal and negative-ideal solutions.

For the members in the Pareto front, we obtain the ideal and negative-ideal solutions according to the thought of Equation (18) and (19). Based on the Euclidean distance of each normalized solution from ideal solution, the closest one is selected as the optimal solution and its corresponding weight values are chosen as the optimal weights.

The Pseudo code of unconstrained-HEFT-T algorithm is shown in Table 3.

C. DEADLINE-CONSTRAINED HEFT-T ALGORITHM

Since the task scheduling problem under MCC environment is formulated as the deadline-constrained multi-objective optimization problem, it is equally important to consider how to process and meet the constraints as well as the objective optimization. Thus, in this section, an adaptive weight adjustment strategy based on the unconstrained HEFT-T algorithm is proposed to cope with this problem.

As it has been mentioned in Section B in IV that the time constraint is treated as an objective along with total cost and mean load, the main idea of this strategy is to increase the weight of time deadline and to decrease the weight values of other two objectives recursively at the same time until the calculated TET satisfy this constraint. It is obvious that the higher weight value of one objective means higher optimization capability for this objective.

Based on the optimal weight value $[w^*_{Time}, w^*_{Cost}, w^*_{load}]$ obtained in unconstrained HEFT-T algorithm, the adjustment for weight value of each objective can be defined as:

$$w_{Time} = w^*_{Time} + \lambda \times \Delta \tag{23}$$

$$w_{Cost} = w^*_{Cost} - 0.5 \times \lambda \times \Delta \tag{24}$$

$$w_{Load} = w^*_{Load} - 0.5 \times \lambda \times \Delta \tag{25}$$

TABLE 4. Pseudo code of deadline constrained HEFT-T algorithm.

Algorithm 3: Deadline constrained HEFT-T algorithm
1. Initialize parameters
2. Compute TET by Algorithm 1
3. **while** $TET > Deadline$
4. **if** $\Delta \leq 0.001$
5. $\Delta = 2$
6. **end**
7. Adjust weight values according to Equation (23), (24) and (25)
8. Normalize the adjusted weight values
9. **if** $w_{time} = 1$
10. break;
11. **end**
12. **while** there are unscheduled tasks in the list **do**
13. Select the first task n_i from the list of scheduling.
14. **for** each processor p_k **do**
15. Compute $EFT(n_i, p_k)$ value according to Equation (5)
16. Compute $Cost_{ik}$ value according to Equation (7) and (8)
17. Compute L_{ik} value according to Equation (9)
18. **end for**
19. Select the optimal provider p_j by applying TOPSIS
20. Assign task n_i to the processor p_j
21. **endwhile**
22. **endwhile**
23. Return total cost, mean load and TET.

TABLE 5. Simulation parameters.

Parameter	n	m	a_1 / a_2	a_3 / a_4	a_5 / a_6	a_7 / a_8	a_9 / a_{10}	a_{11} / a_{12}	a_{13} / a_{14}	Ctt	R	λ
Value	[20,80]	[5,20]	30/100	30/100	300/1000	5/30	10/30	0.1/1	300/1000	0.05	100	0.1

Where Δ is the normalized difference between TET and deadline if it cannot meet this deadline, and λ is the adjustment step which aims to avoid the dramatic change of weights.

Δ reflects the adjustment extent of those objectives, which is given by:

$$\Delta = (TET - Deadline)/TET \tag{26}$$

From Equation (26), we can observe that if TET is much greater than $Deadline$, the weight value of w_{Time} would increase quickly, and then TET would meet the $Deadline$ with a fast speed. Likewise, if the TET is a little beyond the $Deadline$, the weight value of time objective would slowly rise until it satisfies this constraint. This adjustment will stop immediately once TET could satisfy the constraint. The aim of this adjustment is to meet the $Deadline$ while maintaining the optimization of total cost and mean load as the weight values are regulated based on the optimal one obtained in the unconstrained case. However, if the TET is very close to but beyond the deadline, the Δ could be very small, which leads to an extremely slow weight value adjustment. Thus, a condition is added to avoid this case. We let the Δ be set as 2 if it is less than 0.001.

The Pseudo code of deadline constrained HEFT-T algorithm is shown in Table 4.

V. PERFORMANCE EVALUATION

A. EXPERIMENTAL SETTINGS

In order to evaluate the performance of the proposed algorithm, in this section, we evaluate our proposed HEFT-T algorithm with unconstrained and deadline-constrained cases for task scheduling in the MCC environment. The simulations were coded in MATLAB 2016a [23] and is performed on computers with Intel Core i5-4670S CPU (3.10 GHz and 8G RAM).

Various simulation parameters are summarized in Table 5. The number of tasks of an application n is from 20 to 80, and the number of providers m is from 5 to 20. The capacity of memory and computing resources obeys uniform distribution in $[a_1, a_2]$ and $[a_3, a_4]$ respectively. The length of each task (MIPS) submits to the uniform distribution in $[a_5, a_6]$. The resource consumption of memory and computing resource of each task obeys uniform distribution in $[a_7, a_8]$ and $[a_9, a_{10}]$ respectively. The cost of providers per unit time obeys uniform distribution in $[a_{11}, a_{12}]$, and the cost

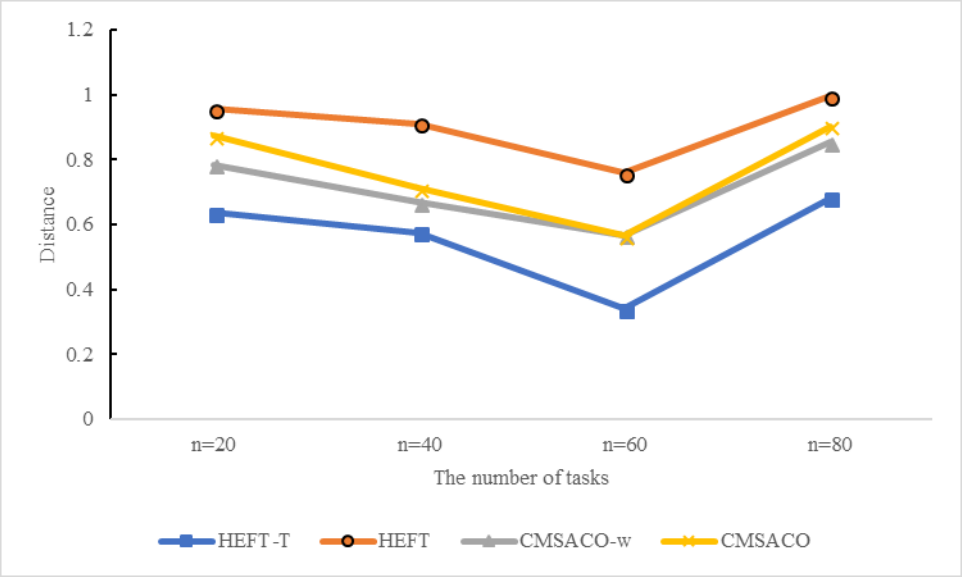

FIGURE 4. Comparison of distances from the calculated solutions to the ideal one obtained by different algorithms.

of per unit time for transmission is set as $Ctt = 0.05$. The amount of data that must be transferred to each task obeys uniform distribution in $[a_{13}, a_{14}]$. The transmission rate is set as $R = 100$. In addition, the capacity of memory and computing resources as well as the cost of providers per unit time are sorted with an ascending order. Thus, in this situation, the cost of the task execution is higher if it is allocated on the provider with high memory and computing capacity. This is more reasonable in the real life.

The DAGs are generated randomly according to the random graph generator [5], which depends on some input parameters, such as the number of tasks in the graph, shape parameter of the graph (height and width) and the out degree of a task node.

B. EXPERIMENTAL RESULTS

We compare our proposed HEFT-T algorithm with HEFT algorithm and Ant Colony Optimization algorithm (CMSACO) [6]. In order to validate the effectiveness of non-dominated sorting in the provider selection phase of our approach, the optimal weight values of objectives $[w^*_{Time}, w^*_{Cost}, w^*_{load}]$ obtained in the evaluation of unconstrained HEFT-T algorithm will be applied in CMSACO. For distinguishing, CMSACO with optimal weight values is named as CMSACO-w, and the weights of objectives in original CMSACO are set as [1/3, 1/3, 1/3]. Moreover, the simulation times for CMSACO-w and CMSACO are set as 10 to get the average of total cost and mean load.

1) EVALUATION OF UNCONSTRAINED HEFT-T ALGORITHM

We analyze the algorithms in terms of total cost and mean load although the time is still considered as an optimization objective in this algorithm. However, it is not easy to judge a set of solutions is better than another set in the multi-objective optimization problem. Thus, we use the distance from the calculated solution to the ideal one to indicate the performance of the results. The shorter of the distance means the closer this solution is to the ideal one.

From Fig. 4, it can be observed that the distance obtained by proposed HEFT-T is the lowest compared with other three algorithms, which means the solutions obtained by our proposed HEFT-T algorithm are much closer to the ideal solutions. With the increase of the number of tasks, our proposed algorithm performs better than those obtained by other algorithms. In addition, we can see that distance obtained by CMSACO-w is shorter than that calculated by CMSACO. It means that the optimal weight values selected in HEFT-T algorithm improves the optimization of total cost and mean load compared with those just giving the same weights to all the objectives.

From Fig. 5, we can see that with the number of the tasks increasing, the total cost rises, and the cost obtained by HEFT-T is still the lowest. While in Fig. 6, we can see that HEFT get the lowest mean load all the time. This is because the provider selection phase in HEFT always aims at selecting the provider with the minimum execution time, and that kind of providers always have higher computing capability as well as resource capacity which leads to low load. For HEFT-T and CMSACO algorithms, it can be seen that for the situation n = 40, both total cost and mean load perform better than that obtained by CMSACO-w and CMSACO. For the situation n = 60, mean load obtained by HEFT-T is slightly higher than that obtained by CMSACO-w while total cost is much lower than that obtained by both CMSACO-w and CMSACO. Therefore, in Fig. 4, it can be observed that the data for distances of

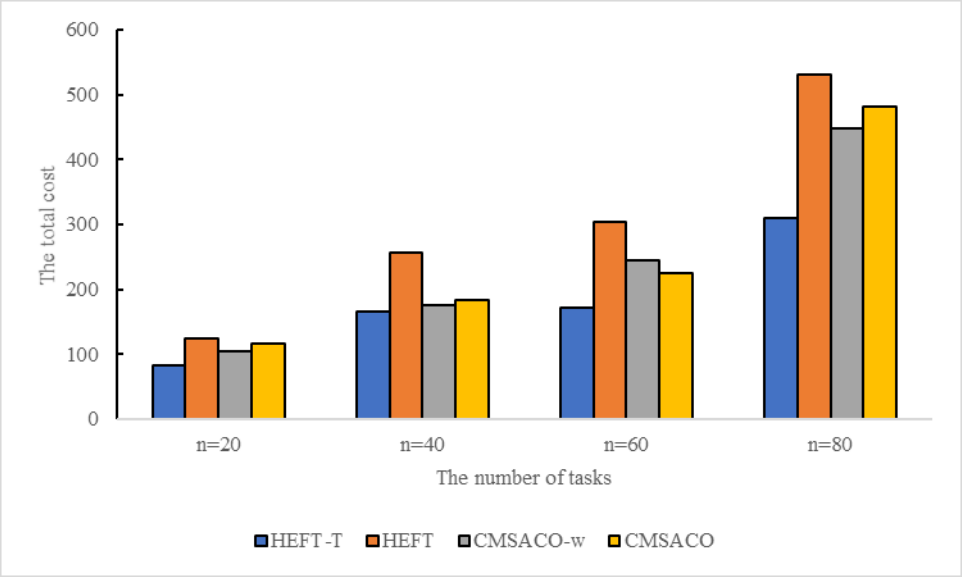

FIGURE 5. Comparison of total cost obtained by different algorithms.

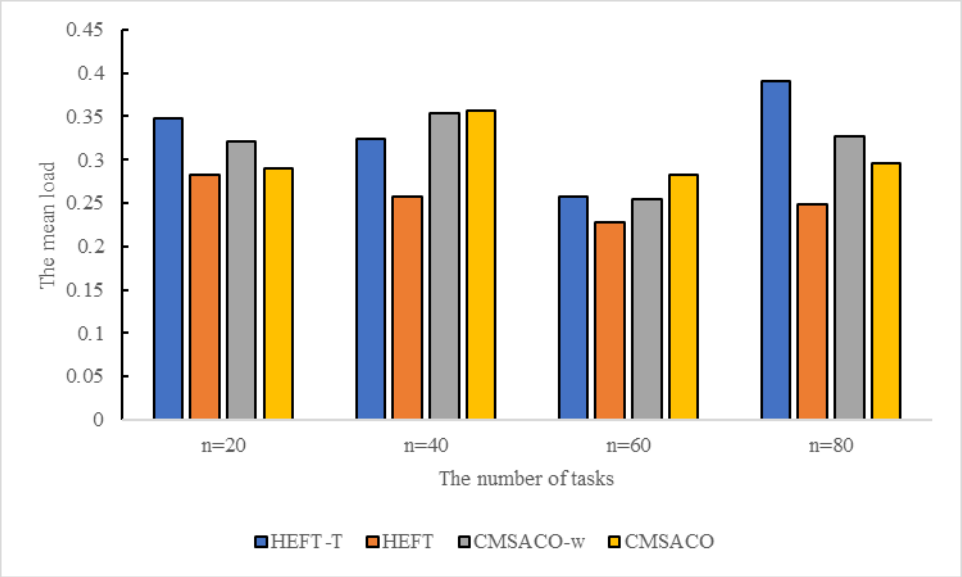

FIGURE 6. Comparison of mean load obtained by different algorithms.

these two situations outperform that of both CMSACO-w and CMSACO. When the number of tasks is 80, the total cost calculated by HEFT-T is much lower than that obtained by other 3 algorithms, but the mean load is higher than others. The main reason of this phenomenon is the setting of experimental parameters. In all the experiments, the provider with high resource capacity and computing capability is with high cost and vice versa. Thus, if the total cost is less than others, which means it mainly selects the providers with lower cost and lower resource capacity to execute those tasks which leads to higher mean load. Furthermore, this is also the reason that we utilize the concept of distance from ideal and negative ideal solutions to measure the performance of the results.

2) EVALUATION OF DEADLINE-CONSTRAINED HEFT-T ALGORITHM

We analyze the average of the TET and the optimization of total cost and mean load for the task scheduling under different deadlines in the MCC environment. There are 4 different deadlines to verify the effectiveness of the proposed algorithm. These deadlines are generated between the slowest and the fastest processing time. As the aim of HEFT

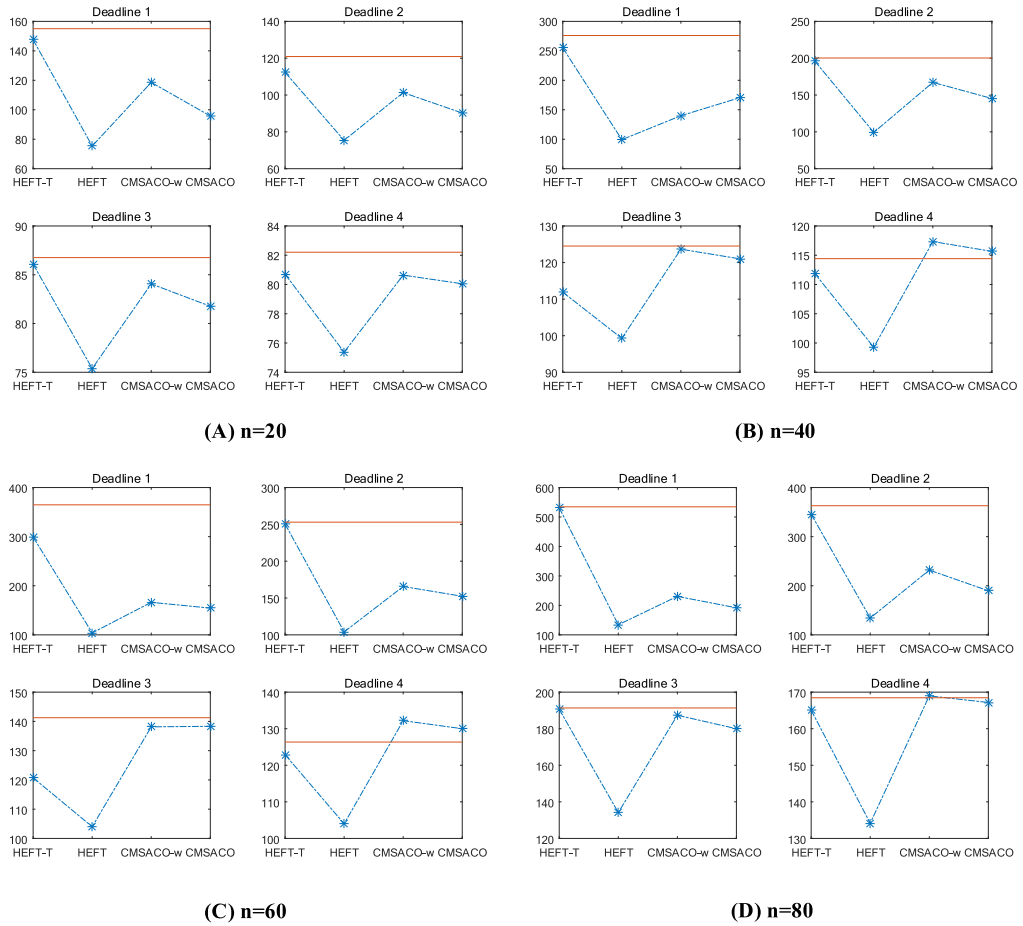

FIGURE 7. Comparison of average TET (dash line) of algorithms under different deadlines (red line).

algorithm is to minimize the TET for the task scheduling, we let this processing time to be the upper limit (the fastest), and the lower limit (the slowest) is calculated by utilizing one provider with average computing capability to execute all the tasks. To estimate each of the 4 deadlines, the difference is divided between the fastest and slowest TET by 10 to obtain an interval size [24]. The first deadline is calculated as the slowest time minus 4 interval sizes. The second is the fastest processing time plus 4 interval sizes. The third is the fastest processing time plus 1 interval sizes, and the last is the fastest processing time plus 0.6 interval sizes.

From the Fig. 7, we can see that HEFT could always be with the minimum TET, and for other algorithms, with the deadline getting stricter, it is harder to meet the deadline. When the number of tasks is set as 20, it can be observed that the mean TET of all the algorithms can satisfy this deadline although it is getting closer to that constraint when it becomes stricter. Because the number of tasks and providers is small, it is easier for all the algorithms to find out the solutions which could meet the deadline compared with more tasks and providers. For the situations of $n = 40$, $n = 60$ and $n = 80$, when the constraint is loose (such as deadline 1 and 2), all the algorithms could satisfy the deadline. Especially for CMSACO-w and CMSACO, they are with the similar optimization capability in the time satisfying. However, when the deadline becomes stricter, the TET obtained by CMSACO-w and CMSACO is closer to the constraint and even cross this line as shown in deadline 4. Compared with the aforementioned results, the TET calculated by proposed HEFT-T still maintains under all the deadlines and all the situations. Thus, it can be seen that our proposed algorithm shows a better performance on deadline satisfying and task scalability.

In addition, it can be noticed that the TET obtained by HEFT-T is very close to but not beyond the deadline compared to other algorithms. This is because the weight value is adjusted based on the optimal weight values computed in the unconstrained case. Thus, in this situation, once the calculated TET could meet the deadline by applying adjusted weight values, the algorithm stops and returns the results immediately. In this way, the optimization ability of total cost and mean load can be retained to some extent.

As shown in Fig. 8, the distance from the ideal solution is calculated by the above algorithms under different deadlines. In addition, the data for total cost, mean load, TET and the meeting rate of deadline are also displayed in Table 6. It is not available for algorithms to obtain

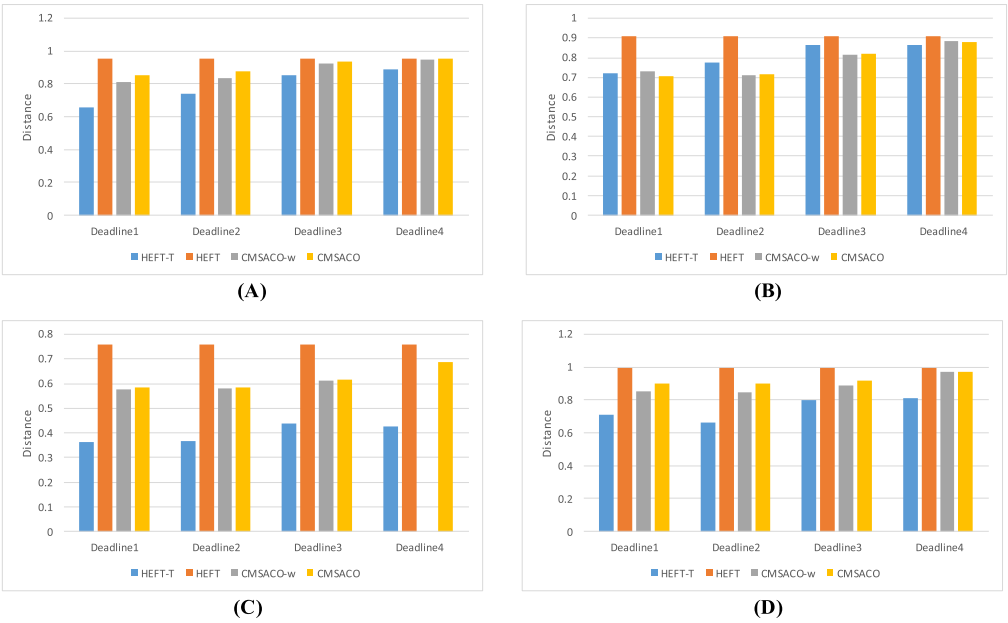

FIGURE 8. Comparison of distances obtained by algorithms under different deadlines.

TABLE 6. Performance comparison of algorithms under different deadlines.

The number of tasks	Algorithms	Total Cost	Mean Load	TET	Distance	Meeting rate (%)	Total Cost	Mean Load	TET	Distance	Meeting rate (%)
		Deadline 1					Deadline 2				
20	HEFT-T	**97.60**	0.30	147.51	**0.65**	100	**105.50**	0.29	112.32	**0.74**	100
	HEFT	123.35	**0.28**	75.38	0.96	100	123.35	**0.28**	75.38	0.96	100
	CMSACO-w	108.14	0.31	118.59	0.81	100	111.72	0.30	101.34	0.84	100
	CMSACO	113.66	0.30	95.78	0.85	100	116.30	0.29	90.24	0.87	100
40	HEFT-T	214.56	0.27	255.06	0.72	100	228.16	**0.26**	196.23	0.77	100
	HEFT	255.78	**0.26**	99.27	0.91	100	255.78	**0.26**	99.27	0.91	100
	CMSACO-w	197.50	0.34	139.90	0.73	100	**184.56**	0.36	167.18	**0.71**	100
	CMSACO	**188.81**	0.35	170.59	**0.71**	100	196.99	0.33	145.34	**0.71**	100
60	HEFT-T	**185.21**	0.25	299.18	**0.36**	100	**188.37**	0.25	250.53	**0.37**	100
	HEFT	303.38	**0.23**	104.00	0.76	100	303.38	**0.23**	104.00	0.76	100
	CMSACO-w	228.31	0.28	166.22	0.58	100	232.53	0.28	165.64	0.58	100
	CMSACO	233.12	0.28	154.47	0.58	100	237.35	0.27	152.34	0.58	100
80	HEFT-T	**287.86**	0.42	531.65	**0.71**	100	**319.52**	0.37	345.77	**0.66**	100
	HEFT	530.98	**0.25**	134.15	1.00	100	530.98	**0.25**	134.15	1.00	100
	CMSACO-w	444.78	0.33	230.64	0.85	100	445.06	0.33	231.84	0.85	100
	CMSACO	482.17	0.29	191.82	0.90	100	484.12	0.29	189.74	0.90	100
		Deadline 3					Deadline 4				
20	HEFT-T	**112.83**	0.31	86.09	**0.85**	100	**114.44**	0.32	80.70	**0.89**	**100**
	HEFT	123.35	**0.28**	75.38	0.96	100	123.35	**0.28**	75.38	0.96	**100**
	CMSACO-w	119.43	0.30	84.06	0.93	100	121.53	0.29	80.64	0.95	90
	CMSACO	121.44	0.29	81.77	0.94	100	121.96	0.30	80.05	0.95	90
40	HEFT-T	247.87	**0.24**	111.90	0.86	100	247.87	**0.24**	111.90	**0.86**	**100**
	HEFT	255.78	0.26	99.27	0.91	100	255.78	0.26	99.27	0.91	**100**
	CMSACO-w	**229.47**	0.30	123.65	**0.81**	100	246.72	0.29	117.34	0.88	20
	CMSACO	232.03	0.29	121.01	0.82	100	**245.54**	0.29	115.66	0.88	20
60	HEFT-T	**209.69**	0.25	120.72	**0.44**	100	**210.00**	0.25	122.90	**0.43**	**100**
	HEFT	303.38	**0.23**	104.00	0.76	100	303.38	**0.23**	104.00	0.76	**100**
	CMSACO-w	249.77	0.27	138.14	0.61	100	-	-	132.20	-	0
	CMSACO	252.18	0.27	138.32	0.62	100	266.59	0.28	130.00	0.68	10
80	HEFT-T	**399.45**	0.37	190.65	**0.80**	100	**429.18**	0.33	165.07	**0.81**	**100**
	HEFT	530.98	**0.25**	134.15	1.00	100	530.98	**0.25**	134.15	1.00	**100**
	CMSACO-w	470.56	0.31	187.37	0.89	100	512.02	0.29	168.98	0.97	30
	CMSACO	491.56	0.29	180.00	0.92	100	511.71	0.29	167.17	0.97	50

the low cost or load if deadlines are not satisfied. Thus, both the total cost and mean load are presented in the case of satisfying the deadline. But the TET is displayed considering the situations of meeting and not meeting the deadline, which aims to verify the performance of deadline meeting.

From Fig. 8, for all the cases with different number of tasks, we can see that the distance from the ideal solution is getting farther with the stricter of the deadline except that of HEFT which is not affected by the change of deadlines. When the number of tasks and providers is small, the optimization ability of total cost and mean load calculated by proposed HEFT-T algorithm is just slightly better than that obtained by other 3 algorithms especially when deadlines get stricter (such as $n = 20$ and $n = 40$). For the situations $n = 60$ and $n = 80$, it can be seen that the distance obtained by HEFT-T is obviously shorter than that computed by other 3 algorithms. We highlight the minimum total cost, mean load and distance with green, blue and red colors respectively in Table 6. It can be noticed that the whole performance of proposed HEFT-T is better than that of other algorithms for most situations.

Moreover, it can be observed that the deadline meeting rates decrease with the stricter of deadlines for a certain number of tasks while scheduling tasks under MCC environment. The reason of this phenomenon is that there are less available solutions when the deadline is getting hard to meet. Meanwhile, we highlight the maximum meeting rate of all the algorithms under deadline 4 (the strictest deadline) with orange color. It can be seen that our proposed HEFT-T and HEFT algorithms can always meet the deadline compared to CMSACO-w and CMSACO.

From the above discussion, the following conclusions can be drawn from the experiments: the CMSACO-w and CMSACO fails to meet deadlines in some cases especially when the deadline is strict, whereas HEFT and HEFT-T can always meet the constraints. Compared with HEFT, CMSACO-w and CMSACO, the proposed HEFT-T algorithm handles better for the constrained multi-objective optimization problem by considering the distance from the ideal deadline.

VI. CONCLUSIONS

In this paper, a HEFT applied TOPSIS (HEFT-T) algorithms is proposed for the unconstrained and deadline constrained task scheduling in MCC environments. Most works focus on single objective optimization and did not concentrate on how to highlight the processing of constraints when the task scheduling is modeled as a constrained optimization problem. Thus, our proposed algorithm aims to address these problems by obtaining the optimal solutions through a three-stage strategy under unconstrained problem and adjusting the weight values for time and other objectives adaptively in order to satisfy the deadline under the time constrained case. Experimental results show that our proposed algorithm achieves better performance on the optimization of total cost as well as mean load and meets the deadlines under strict constraints while the CMSACO algorithm could not succeed easily.

REFERENCES

[1] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2013, doi: 10.1109/SURV.2013.050113.00090.

[2] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2013, doi: 10.1109/SURV.2013.062613.00160.

[3] B. G. Chun and P. Maniatis, "Augmented smartphone applications through clone cloud execution," in *Proc. Conf. HotOS*, Monte Verità, CA, USA, 2009, pp. 1–8.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: Architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, Dec. 2013, doi: 10.1002/wcm.1203.

[5] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260–274, Mar. 2002, doi: 10.1109/71.993206.

[6] T. Wang, X. Wei, C. Tang, and J. Fan, "Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints," *Peer-to-Peer Netw. Appl.*, vol. 11, no. 4, pp. 793–807, 2018, doi: 10.1007/s12083-017-0561-9.

[7] L. Shakkeera and L. Tamilselvan, "QoS and load balancing aware task scheduling framework for mobile cloud computing environment," *Int. J. Wireless Mobile Comput.*, vol. 10, no. 4, pp. 309–316 Jan. 2016, doi: 10.1504/IJWMC.2016.078201.

[8] H. Liu *et al.*, "A holistic optimization framework for mobile cloud task scheduling," *IEEE Trans. Sustain. Comput.*, pp. 1–14, Oct. 2017. doi: 10.1109/TSUSC.2017.2765520.

[9] C. L. Hwang and K. Yoon, *Multiple Attribute Decision Making*. Berlin, Germany: Springer, 1981.

[10] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Proc. Conf. PPSN VI*, Paris, France, 2000, pp. 849–858.

[11] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control in market-oriented environments," in *Proc. GECON*, Paphos, Cyprus, 2012, pp. 105–119.

[12] H. Arabnejad and J. G. Barbosa, "A budget constrained scheduling algorithm for workflow applications," *J. Comput.*, vol. 12, no. 4, pp. 665–679, Dec. 2014, doi: 10.1007/s10723-014-9294-7.

[13] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Comput.*, vol. 17, no. 2, pp. 169–189, Jun. 2013, doi: 10.1007/s10586-013-0325-0.

[14] Y. Li, M. Chen, W. Dai, and M. Qiu, "Energy optimization with dynamic task scheduling mobile cloud computing," *IEEE Syst. J.*, vol. 11, no. 1, pp. 96–105, Mar. 2017, doi: 10.1109/JSYST.2015.2442994.

[15] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 175–186, Dec. 2014, doi: 10.1109/TSC.2014.2381227.

[16] P. P. Hung and E.-N. Huh, "An adaptive procedure for task scheduling optimization in mobile cloud computing," *Math. Problems Eng.*, vol. 2015, pp. 1–14, 2015, doi: 10.1155/2015/969027.

[17] X. Wei, J. Fan, Z. Lu, and K. Ding, "Application scheduling in mobile cloud computing with load balancing," *J. Appl. Math.*, vol. 2013, no. 1, pp. 1–13, Nov. 2013, doi: 10.1155/2013/409539.

[18] P. P. Hung, T. A. Bui, and E. N. Huh, "A new approach for task scheduling optimization in mobile cloud computing," in *Frontier and Innovation in Future Computing and Communications* (Lecture Notes in Electrical Engineering). Dordrecht, The Netherlands: Springer, 2014, ch. 26, pp. 211–220.

[19] S. E. Mahmoodi, R. N. Uma, and K. P. Subbalakshmi, "Optimal joint scheduling and cloud offloading for mobile applications," *IEEE Trans. Cloud Comput.*, pp. 1–27, Apr. 2016. doi: 10.1109/TCC.2016.2560808.

[20] H. Shah-Mansouri, V. W. Wong, and R. Schober, "Joint optimal pricing and task scheduling in mobile cloud computing systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5218–5232, Aug. 2017, doi: 10.1109/TWC.2017.2707084.

[21] Z. Cai and C. Chen, "Demand-driven task scheduling using 2D chromosome genetic algorithm in mobile cloud," in *Proc. PIC*, Shanghai, China, 2014, pp. 539–545.

[22] S. Ghasemi-Falavarjani, M. Nematbakhsh, and B. S. Ghahfarokhi, "Context-aware multi-objective resource allocation in mobile cloud," *Comput. Elect. Eng.*, vol. 44, pp. 218–240, May 2015, doi: 10.1016/j.compeleceng.2015.02.006.

[23] (2018). *MathWorks*. [Online]. Available: http://www.mathworks.com/

[24] L. Liu, M. Zhang, R. Buyya, and Q. Fan, "Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing," *Concurrency Computat. Pract. Exper.*, vol. 29, no. 5, pp. 1–12, Mar. 2017, doi: 10.1002/cpe.3942.

[25] X. Lin, Y. Wang, Q. Xie, and M. Pedram, "Energy and performance-aware task scheduling in a mobile cloud computing environment," in *Proc. CLOUD*, Washington, DC, USA, 2014, pp. 192–199.
[26] Y. Xia, M. Zhou, X. Luo, S. Pang, and Q. Zhu, "Stochastic modeling and performance analysis of migration-enabled and error-prone clouds," *IEEE Trans Ind. Informat.*, vol. 11, no. 2, pp. 495–504, Apr. 2015, doi: 10.1109/TII.2015.2405792.
[27] Y. Xia, M. C. Zhou, X. Luo, Q. Zhu, J. Li, and Huang, Y, "Stochastic modeling and quality evaluation of infrastructure-as-a-service clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 162–170, Sep. 2013, doi: 10.1109/TASE.2013.2276477.
[28] Q. Wu, M. C. Zhou, Q. Zhu, and Y. Xia, "VCG auction-based dynamic pricing for multigranularity service composition," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 796–805, Apr. 2017, doi: 10.1109/TASE.2017.2695123.
[29] Q. Wu, F. Ishikawa, Q. Zhu, Y. Xia, and J. Wen, "Deadline-constrained cost optimization approaches for workflow scheduling in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3401–3412, Aug. 2017, doi: 10.1109/TPDS.2017.2735400.
[30] J. Liu, Y. Mao, J. Zhang, and K. B. Letaief, "Delay-optimal computation task scheduling for mobile-edge computing systems," in *Proc. ISIT*, Barcelona, Spain, 2016, pp. 1451–1455.

LI LIU received the Ph.D. degree from the University of Science and Technology Beijing, China, in 2006. She is currently a Professor with the School of Automation and Electrical Engineering, University of Science and Technology Beijing. Her research interests are in the area of service composition and resource allocation in the cloud computing and mobile cloud computing.

QI FAN received the B.S. degree in automation from the University of Science and Technology Beijing, Beijing, China, in 2016, where she is currently pursuing the master's degree in control science and engineering. Her research interest includes the resource allocation and task scheduling in the mobile cloud computing environment.

RAJKUMAR BUYYA received the Ph.D. degree in computer science and software engineering from Monash University, Melbourne, Australia, in 2002. He is currently a Professor of computer science and software engineering and the Director of the Cloud Computing and Distributed Systems Laboratory, The University of Melbourne, Australia. His research interests are in the area of cloud computing, mobile cloud computing, big data, and Internet of things.