The Journal of Supercomputing (2023) 79:1814–1833
https://doi.org/10.1007/s11227-022-04681-3

A cost and makespan aware scheduling algorithm for dynamic multi-workflow in cloud environment

Yuanqing Xia[1] · Yufeng Zhan[1,2] · Li Dai[1] · Yuehong Chen[1]

Accepted: 20 June 2022 / Published online: 2 August 2022
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2022

Abstract

With the development of cloud computing, a growing number of workflows are deployed on cloud platform that can dynamically provides cloud resources on demand for users. In clouds, one basic problem is how to schedule workflow for minimizing the execution cost and the workflow completion time. Aiming at the problem that the maximum completion time and cost of multiple workflows are too high, this paper proposes a model of dynamic multi-workflow scheduling in cloud environment and a new scheduling algorithm which is named as MT (multi-workflow scheduling technology). In MT, the heterogeneity of resources is considered when calculating the priority of tasks. Then, the technique for order of preference by similarity to ideal solution (TOPSIS) method is used to rank the resources when selecting resources for tasks. Finally, MT takes the estimated minimum completion time of the workflow and the cost of the task as two attribute indexes in TOPSIS decision matrix. Also, it uses a fixed reference point instead of calculating ideal solution, which ensures the uniqueness of the evaluation criteria when there is a change in the number of resources. Simulation experiments are illustrated to verify the effectiveness of the proposed algorithm in reducing the maximum completion time and cost of multiple workflows. Compared with the state-of-the-art methods, the maximum completion time and cost can be reduced by at most 17 and 9%, respectively.

Keywords Cloud computing · Multi-workflow scheduling · Cost · Makespan

✉ Yufeng Zhan
yu-feng.zhan@bit.edu.cn

Yuanqing Xia
xia_yuanqing@bit.edu.cn

Li Dai
li.dai@bit.edu.cn

Yuehong Chen
yuehchen@163.com

1 School of Automation, Beijing Institute of Technology, Beijing 100081, China

2 Yangtze Delta Region Academy of Beijing Institute of Technology, Jiaxing 314000, China

1 Introduction

With the rapid development of cloud computing technology, the cloud resource provides an efficient computing service model and provided great convenience for human life [1, 2] and is widely used in complex applications with the powerful computing capability [3], such as transportation, medical care, education and e-commerce industries [2]. In clouds, the cloud model consists of a great number of servers which are equipped with adequate cloud resources, such as CPU cores and memory and multiple virtual machines (VMs) instances are running simultaneously on these servers. In this way, many workflows applications are executed in cloud environment. As a result, there are many challenges for cloud service providers how to effectively schedule applications with cloud resources [4].

Workflow scheduling in cloud computing refers to obtaining the corresponding time and space mapping relationship between tasks and resources [5] and allocating tasks to proper resources according to different scheduling objectives, which not only plays a decisive role in the whole cloud workflow system, but also greatly affects QoS requirements of users [6]. Many heuristics or meta-heuristic algorithms [7–9] have been proposed for workflow scheduling to optimize a single objective, such as the scheduling length or execution cost. However, more than one objective need to be taken into consideration. Time and cost are the two most important but conflict QoS parameters, which increases the difficulty of workflow scheduling. In addition, current approaches mainly focuses on single workflow scheduling. As for multiple workflows, they will schedule the workflows sequentially, which cannot extract all the features of workflows so as to give the optimal scheduling strategy. In the cloud, the resources with the best performance usually have the most expensive prices. Therefore, how to balance these two parameters in scheduling when multiple workflow dynamically arrive is a challenge.

In this paper, we design a scheduling and optimization algorithm for dynamic scheduling multiple workflows in clouds. Different from current approaches, the proposed approach can schedule multiple workflows simultaneously, and the objective of this paper is to minimize the maximum completion time and the total cost for executing the dynamic multiple workflows. The main contributions are as follows:

- We consider the heterogeneity of resources when calculating the priority of tasks. And the TOPSIS[1] method is adopted to select resources for tasks.
- A new algorithm called MT is proposed, which can minimize the maximum completion time and the total cost of the multiple worklflows. Considering the influence of resource selection on the completion time of the child tasks, the estimated minimum completion time of the workflow is applied as one attribute index, and the sum of the tasks' execution cost and data transmission cost is

[1] The Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) is a multi-criteria decision analysis method, it is based on the concept that the chosen alternative should have the shortest geometric distance from the positive ideal solution and the longest geometric distance from the negative ideal solution [10].

taken as the other attribute index, which can further reduce the execution time and cost of the workflow. In addition, it adopts a fixed reference point instead of calculating ideal solution, which ensures the uniqueness of the evaluation criteria when there has a change in the number of resources.

- Simulation experiments are carried out to verify the effectiveness of the proposed algorithm. Experimental results validate that the proposed MT algorithm can generated less maximum completion time and total cost of multiple workflows than the state-of-the-art algorithms.

The remainder of this paper is organized as follows. Section 2 presents the related work. System model and the scheduling problem are given in Sect. 3. The proposed MT algorithm is introduced in Sect. 4. In Sect. 5, experiments are carried out to evaluate the algorithm's efficiency. Finally, Sect. 6 concludes the paper.

2 Related work

There are many researches on workflow scheduling. And they can be divided into several categories according to different optimization objectives, such as cost or scheduling length. The heterogeneous earliest-finish-time (HEFT) algorithm has been proposed in [7], which is a heuristic algorithm to minimize the scheduling length of the workflow in the heterogeneous environment. Based on this, Lin et al. [11] proposed scalable heterogeneous earliest-finish-time (SHEFT) algorithm, which successfully realized the elastic scaling of resources in the process of scheduling, and effectively reduced the execution time of tasks. Besides the heuristic algorithms, some scholars use meta-heurisitc algorithms to slove these problems. Buyya [12] proposed particle swarm optimization (PSO) to generate the schedule plan. And [13] adopted ant colony optimization (ACO) algorithm to schedule workflows. For optimizing the execution cost and time, Pareto optimal scheduling heuristic (POSH) as a multi-objective heuristic optimization algorithm has been proposed by Su [14] in 2013, which aims to achieve the balance between the execution cost and the maximum completion time of the workflow. It selects tasks in the same way as described in HEFT. In the stage of resource selection, the task execution cost and execution time are weighted respectively and added to obtain a new objective function. Based on the new objective function, the optimal resource for the task can be selected.

Recently, Li et al. considered a multiobjective workflow scheduling problem and proposed a scoring and dynamic hierarchy-based NSGA-II (nondominated sorting genetic algorithm II), to minimize both workflow makespan and cost [15]. Chen et al. [16] adopted co-evolutionary multiple populations to design a novel multi-objective workflow scheduling algorithm with the ant colony system to minimize both workflow execution time and cost. Zhu et al. [17] proposed an evolutionary multiobjective optimization (EMO)-based algorithm with novel schemes for fitness evaluation, genetic operator. Garg and Singh [18] designed a multiobjective workflow scheduling optimization approach to optimize makespan and monetary cost simultaneously for the whole workflow execution with designed genetic operations in hybrid clouds.

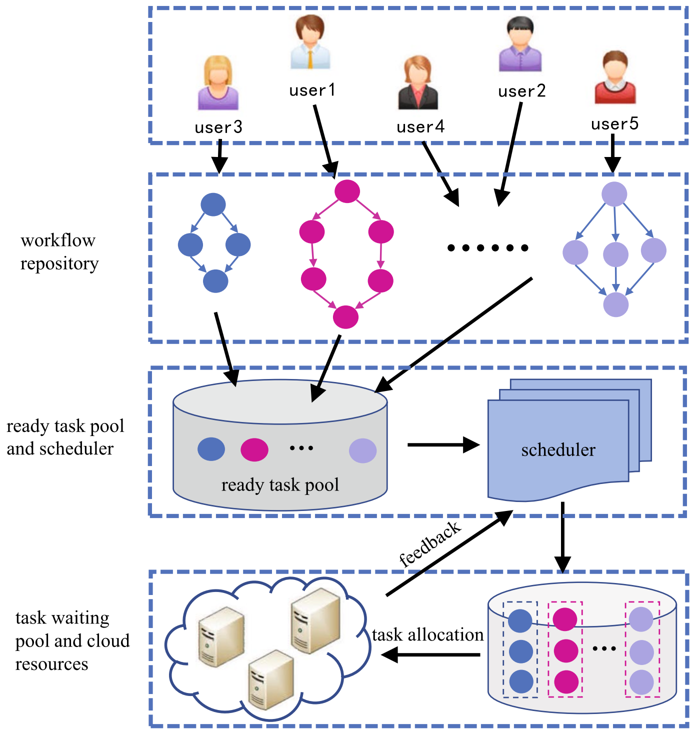

Fig. 1 The dynamic scheduling model of multiple workflows

However, previous studies are only focused at the single workflows. Aiming at the multiple workflows, the online workflow management (OWM) algorithm designed by Hsu et al. [19] to solve the scheduling problem of multiple and hybrid parallel workflows. However, because only idle resources are considered, the task may be delayed, which causes the increase of the completion time of the workflow. Different from current works, the proposed approach in this paper focuses on the workflow scheduling which has multi-objectives. In addition, our approach can schedule the multiple workflows simultaneously, which will match the optimal resources for each task, so as to make full use of the resources.

3 System model

The dynamic multi-worklfow model is illustrated in Fig. 1. After accepting the application requests, the system abstract them into the directed acyclic graph (DAG) and store them in the workflow repository. Ready task pool refers to tasks that are fully prepared and can be scheduled at any time in the workflow repository according to certain selection rules, such as round robin or first come first schedule. Various scheduling algorithms are embedded in the scheduler, which can intelligently select the optimal scheduling algorithms to assign tasks to appropriate resources according to the users' personalized needs, and obtain the corresponding mapping relationship between tasks and resources.The task waiting queue mainly stores the corresponding waiting tasks on each resource. Finally, the specific scheduling is implemented in the cloud environment according to the scheme given in the scheduler and the order of tasks in the task waiting queue. The usage status of related cloud resources in the cloud also needs to be feed back to the scheduler in real time to provide real-time information for schedule subsequent tasks. In this paper, when a resource in the cloud resource pool executes a task, it can only process one task at a time, in other words, the execution of the task can't be preempted.

Cloud service provider such as Amazon cloud can provide a platform which comprises a large number of virtual machines (VMs) or containers with different types. The VMs and containers in the cloud environment are collectively referred to as resources in the paper. The cloud environment is heterogeneous, that is, there are differences in computing and storage capacities among the cloud resources. And let $P = \{p_1, p_2, \dots, p_{|P|}\}$ denote the cloud resources provided, and $|P|$ is the size of the resources. Traditionally, the applications submitted by different users could be modeled as workflows and denoted as $GS = \{G_1, G_2, \dots, G_{|M|}\}$, where $|M|$ is the number of the workflows. Each workflow can be decomposed into many subtasks with precedence constraints. In this paper, the workflow is abstracted as a DAG and denoted as $G(T, E, W, C)$, where $T = \{t_1, t_2, \dots, t_{|N|}\}$ is the set of $|N|$ tasks, and E is the set of communication edges, where $E = \{e_{ij} \| i, j = 1, \dots |N|\}$ represents the data dependency constraints set for tasks of workflow. The $c_{ij} \in C$ is the data transmission time between t_i and t_j. W is the $|N| \times |P|$ computation matrix, and $|N|$ is the number of tasks in the DAG and $|P|$ is the number of resources provided. $w_{i,k} \in W$ represents the time of task t_i processed on resource p_k. The workflow model based on DAG is shown in Fig. 2 and the corresponding computation matrix is shown in Fig. 3.

Next, some definitions of the DAG model are described as follows:

- $pred(t_i)$: The set of the parent tasks of task t_i. In the Fig. 2, task t_1 is the parent task of task t_2, t_3, t_4, t_5 and t_6. The task set composed of task t_2, t_4 and t_6 is the parent task set of task t_8, and the set composed of task t_2, t_4 and t_5 is the parent task set of task t_9. If a task has no parent task, the task is called the entry task of the DAG. If a DAG has more than one entry task, for convenience, a dummy entry task is added in the beginning of the DAG, which has zero execution time and zero–weight with the actual entry task.
- $succ(t_i)$: The set of the child tasks of task t_i. In the Fig. 2, the set composed of task t_2, t_3, t_4, t_5 and t_6 is the child task set of t_1, and the task set consist of task t_8 and t_9 is the child task set of task t_2. If a task's child task set is empty, the task is called the exit task of the DAG. If an DAG has more than one exit task, for convenience, a dummy exit task is added in the ending of the DAG, which has zero execution time and zero–weight with the actual exit task of the DAG.
- $\mathrm{EST}(t_i, p_k)/\mathrm{EFT}(t_i, p_k)$: The earliest start/finish time of task t_i in resource p_k.
- $\mathrm{AST}(t_i)/\mathrm{AFT}(t_i)$: The actual start/finish time of task t_i based on the actual scheduling.
- $\mathrm{Makespan}(G_m)$: The finish time for the workflow G_m. The finish time of G_m can be calculated by:

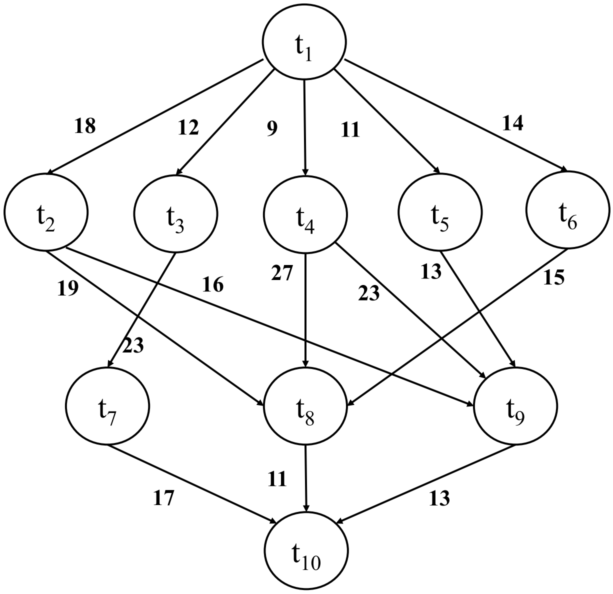

Fig. 2 The DAG model of workflow

task	p_1	p_2	p_3
t_1	14	16	9
t_2	13	19	18
t_3	11	13	39
t_4	13	8	17
t_5	12	13	10
t_6	13	16	9
t_7	7	15	11
t_8	5	7	14
t_9	18	12	20
t_{10}	21	7	16

Fig. 3 The corresponding computation matrix in Fig. 2

$$\mathrm{Makespan}(G_m) = \mathrm{AFT}(G_m.t_{\mathrm{exit}}) - \mathrm{AST}(G_m.t_{\mathrm{entry}}), \tag{1}$$

where $\mathrm{AFT}(G_m.t_{\mathrm{exit}})$ indicates the actual finish time of the exit task of G_m and $\mathrm{AST}(G_m.t_{\mathrm{entry}})$ is the actual start time of the entry task of G_m.

3.1 Cost model

Cloud provides a strategy of pay-as-you-go pricing, where users are charged according to the used time of the resources. Owning to the various capacities of the resources, their prices are also different. Resources with rapid processing speed are expensive, and cheaper resources always have slow processing capacities. In the paper, task's cost consists of computation cost and transmission cost. We use $\mathrm{price}(p_k)$ to denote the price per unit time associated with processor p_k and $\mathrm{price}(p_t)$ to represent the price of data transmission in unit time. The computation and transmission cost of task t_i is

$$C(t_i, p_k) = \mathrm{price}(p_k) * w_{i,k} + \sum_{t_m \in pred(t_i)} c_{mi} * \mathrm{price}(p_t), \tag{2}$$

The cost of workflow G_m is the sum of actual cost of all its tasks

$$C(G_m) = \sum_{t_i \in G_m} C(t_i, p_k). \tag{3}$$

3.2 Problem formulation

Based on the system scheduling model and the cost model, the scheduling problem in this paper is to search the proper map from tasks to resources to minimize the maximum completion time and cost in the multi-workflow system when multiple workflows arrive dynamically, which is

$$\begin{aligned} &\text{minimize} \quad \mathrm{Makespan}(\mathrm{GS}), \\ &\text{minimize} \quad C(\mathrm{GS}), \end{aligned} \tag{4}$$

where Makespan(GS) represents the maximum completion time of all workflows in GS, i.e., the total time from the beginning of the first task to the completion of last task in GS. C(GS) represents the total cost of all workflow. Makespan(GS) and C(GS) are calculated via

$$\begin{aligned} \mathrm{Makespan}(\mathrm{GS}) &= \max_{G_m \in \mathrm{GS}} \max_{t_{\mathrm{exit}} \in G_m} \mathrm{AFT}(t_{\mathrm{exit}}) \\ C(\mathrm{GS}) &= \sum_{G_m \in \mathrm{GS}} C(G_m) \end{aligned} \tag{5}$$

4 Minimize the maximum completion time and cost of multi-workflows

In this section, an algorithm named MT based on TOPSIS method is proposed for minimizing the maximum completion time and cost for multi-workflows. The algorithm mainly contains two phases, which are the task selection and the processor selection based on TOPSIS method. The multi-workflow scheduling process by MT algorithm is shown in Fig. 4. These two phases are described in detail as follows.

4.1 Task selection

The traditional method of calculating tasks' priorities, such as the average execution time of tasks on resources when calculating $rank_u$ in HEFT [7] algorithm, eliminates the heterogeneity among resources' performance. In this section, we consider the heterogeneity of resources in the cloud computing when calculating the priorities of tasks and iteratively calculate the rank value of tasks on each resource in turn

$$\begin{cases} \mathrm{Rank}_n(t_i, p_k) = w_{i,k} + \max\limits_{t_j \in succ(t_i)} \{c_{ij} + \mathrm{Rank}_n(t_j, p_k)\}, \\ \mathrm{Rank}_n(t_{\mathrm{exit}}, p_k) = w_{\mathrm{exit,k}}, \end{cases} \tag{6}$$

where $\mathrm{Rank}_n(t_i, p_k)$ represents the longest distance between task t_i and the exit task when resource p_k is selected by task t_i. $w_{i,k}$ and $w_{\mathrm{exit,k}}$ indicate the execution time of task t_i and t_{exit} on resource p_k, respectively. c_{ij} is the time of data transmission between task t_i and its child task t_j.

According to equation (6), the priority of task t_i can be determined by

$$\mathrm{Rank}_n(t_i) = \sum_{p_k \in P} \mathrm{Rank}_n(t_i, p_k)/|P|. \tag{7}$$

After obtaining the value of Rank_n for all tasks in each workflow, place the tasks in the *init_queue* of the corresponding workflow in the order of decreasing Rank_n. In order to ensure fairness, for the unscheduled workflows, the tasks with the largest Rank_n in every workflow's *init_queue* are submitted to the *ready_pool*, and then the tasks in the *ready_pool* are reordered according to the following formula

$$\mathrm{Rank}_r(G_m \cdot t_i) = \frac{1}{\mathrm{PRT}(G_m)} + \frac{1}{\mathrm{CPL}(G_m)}, \tag{8}$$

where $\mathrm{PRT}(G_m)$ represents the percentage of the remaining unscheduled tasks in the workflow G_m, and $\mathrm{CPL}(G_m)$ is the critical length of G_m. The equation indicates that if there are two workflows with the same number of tasks, the task of the workflow with the less unscheduled tasks and the smaller CPL will gets a high priority.

After calculating the Rank_r of all tasks in the *ready_pool*, selecting the task t_{curr} with the largest Rank_r value, and determining the appropriate resources for

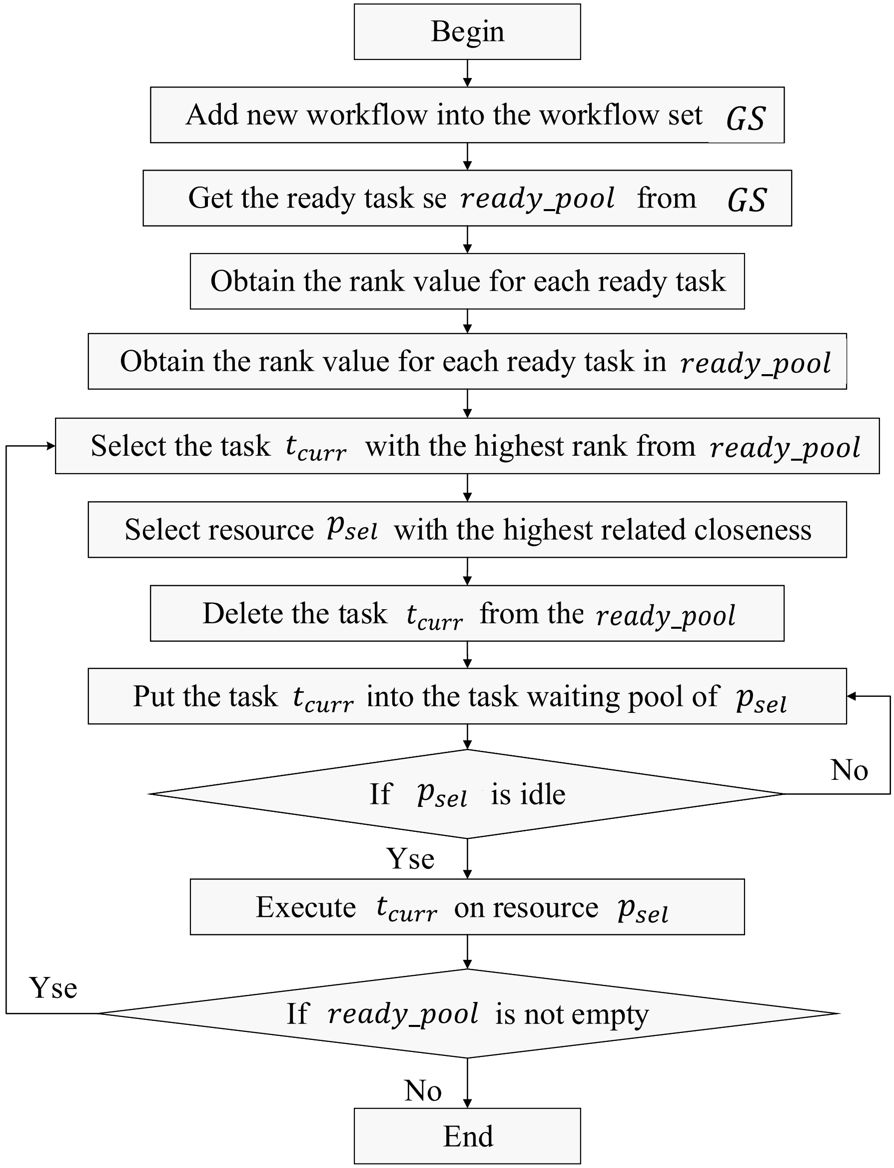

Fig. 4 The multi-workflow scheduling process by MT algorithm

the selected task according to TOPSIS method. Before introducing the strategy of resource selection, next we first introduce TOPSIS method for multi-attribute decision-making.

4.2 Overview for TOPSIS method

TOPSIS [20] method for multi-attribute decision-making is simple in calculation and rigorous in logic, which can intuitively show the gap among the various schemes. The main steps of the TOPSIS method is

(1) Based on the evaluation index of the problem and the given multiple optional schemes, the original decision matrix X of the problem is determined via

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}, \quad (i = 1, 2, \dots, m; j = 1, 2, \dots, n), \tag{9}$$

where x_{ij} is the value of the jth evaluation index on the ith scheme, m is the number of given optional schemes, and n is the number of attribute indexes.

(2) Because the attribute indexes to be evaluated are different from each other, the dimensions of data in decision matrix X are also quite different. In order to eliminate the influence of data dimension, this section uses vector method to standardize the data of X as follows

$$q_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} {x_{ij}}^2}} \quad (i = 1, 2, \dots, m;\ j = 1, 2, \dots, n). \tag{10}$$

After the standardization of X, the dimensionless matrix Q is

$$Q = \begin{bmatrix} q_{11} & q_{12} & \cdots & q_{1n} \\ q_{21} & q_{22} & \cdots & q_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ q_{m1} & q_{m2} & \cdots & q_{mn} \end{bmatrix}, \quad (i = 1, 2, \dots, m;\ j = 1, 2, \dots, n). \tag{11}$$

(3) In the actual problem, the significance of different attribute indexes are various. Thus, it is necessary to determine the weight of attribute indexes in advance according to the actual demand, and to multiply the determined index weight by the corresponding value in Q to obtain the weighted matrix V as

$$v_{ij} = q_{ij} * w_j, \quad \sum_{j=1}^{n} w_j = 1, \quad (i = 1, 2, \dots, m;\ j = 1, 2, \dots, n). \tag{12}$$

(4) Determining the positive ideal solution Z^+ and negative ideal solution Z^- in matrix V, and for different types of indexes, Z_j^+ and Z_j^- can be expressed as follows, respectively,

$$Z_j^+ = \begin{cases} min(v_{ij}), & \text{index of minimization,} \\ max(v_{ij}), & \text{index of maximization,} \end{cases} \\ (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n), \tag{13}$$

$$Z_j^- = \begin{cases} max(v_{ij}), & \text{index of minimization,} \\ min(v_{ij}), & \text{index of maximization,} \end{cases} \\ (i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n). \tag{14}$$

(5) Determining the Euclidean distance between scheme i and positive ideal solution Z^+ and negative ideal solution Z^- as

$$\begin{aligned} d_i^+ &= \sqrt{\sum_{j=1}^{n} (Z_j^+ - v_{ij})^2}, \\ d_i^- &= \sqrt{\sum_{j=1}^{n} (Z_j^- - v_{ij})^2}, \\ &(i = 1, 2, \ldots, m;\ j = 1, 2, \ldots, n). \end{aligned} \tag{15}$$

(6) The relative closeness R_i between the scheme i and the ideal solution Z^+ is

$$R_i = \frac{d_i^-}{d_i^+ + d_i^-}, \quad (i = 1, 2, \ldots, m). \tag{16}$$

It can be seen from equation (16) that the closer R_i is to 1, the closer the scheme is to the positive ideal solution Z^+, in other words, the better the scheme i is compared with other schemes. Each candidate scheme is sorted according to the principle of the relative closeness decreasing.

(7) According to the practical problems and the sorting results of all schemes, the best scheme can be selected.

4.3 Resource selection based on TOPSIS method

Combined with the actual scheduling problem of this paper, this section uses TOPSIS method to select th most appropriate resource for the task t_{curr}. The main steps is:

Step1: Deterimine the original decision matrix X_{m*n}. The number m of X_{m*n} is the size $|P|$ of the set of provided resources. The objective of the paper is to minimize the completion time and cost of the workflows. Thus, the number n is 2. When the task t_{curr} is executed on the resource p_i, the EFT(t_{curr}, p_i) of t_{curr} on the resource p_i is firstly calculated by using formula (17) and the resource insertion strategy is

$$\begin{aligned} \text{EST}(t_{\text{curr}}, p_i) &= \max\{T_{avail}(p_i), \max_{t_m \in pred(t_{\text{curr}})} (\text{AFT}(t_m) + c_{m(\text{curr})})\}, \\ \text{EFT}(t_{\text{curr}}, p_i) &= \text{EST}(t_{\text{curr}}, p_i) + w_{\text{curr},i}, \end{aligned} \tag{17}$$

where $T_{\text{avail}}(p_i)$ is the available time of resource p_i, and the inner max of $\text{EST}(t_{\text{curr}}, p_i)$ indicates that only if all parent tasks of t_{curr} have been finished and the data required by t_{curr} have arrived at p_k, then t_{curr} can be ready for processing.

When optimizing the completion time of the workflow, we should not only consider the impact of resource selection on the current task's completion time, but also take the influence on the task completion time of the child task. Therefore, calculating the maximum value of the shortest path from all child tasks of t_{curr} to the exit task is

$$\text{Min}(t_{\text{curr}}, p_i) = \max_{t_s \in \text{succ}(t_{\text{curr}})} \{\min_{p_o \in P}\{w_{s,o} + c_{(\text{curr})s} + \text{Min}(t_s, p_o)\}\}, \tag{18}$$

where $w_{s,o} + c_{(\text{curr})s}$ represents the sum of the computation time of the child task t_s on resource p_o and the data transmission time between t_s and t_{curr} when t_{curr} selects p_i. Note that, when t_{curr} and t_s select the same resource, the data transmission time is $c_{(\text{curr})s} = 0$.

Adding $\text{Min}(t_{\text{curr}}, p_k)$ to the calculated $\text{EFT}(t_{\text{curr}}, p_i)$ to estimate the minimum completion time of the workflow $\text{Min}G(t_{\text{curr}}, p_i) = \text{EFT}(t_{\text{curr}}, p_i) + \text{Min}(t_{\text{curr}}, p_i)$. Making $\text{Min}G(t_{\text{curr}}, p_i)$ as the first attribute index of the decision matrix, namely, $x_{i1} = \text{Min}G(t_{\text{curr},p_i})$.

The second attribute index of the decision matrix is the sum of the computation cost and the data transmission cost of t_{curr} is defined as $x_{i2} = C(t_{\text{curr},p_i})$.

step2: The dimensionless matrix Q_{m*n} is obtained by standardizing X_{m*n} according to the equation (10).

step3: According to the equation (12) and the weight determined for the two attribute indexes, we can get the weighted matrix V_{m*n}. In this paper, we assume that the completion time is more significant than the cost, so we set the weight of time and cost to 0.9 and 0.1, respectively.

step4: When the traditional TOPSIS method matches resources for tasks, if the size of the given resource increases or decreases, it is likely that the ideal solution will change, which may cause that the evaluation results of the same two resources will reverse. In order to solve this problem, we use a fixed reference point instead of determining ideal solution (Sect. 4.2 (4)), which can ensure the uniqueness of evaluation criteria and the robustness of the proposed algorithm under different size of resources.

Since the two attribute indexes in this section belong to the indexes of minimization and , We use 0 as the positive ideal solution and 1 as the negative ideal solution as follows

$$\begin{aligned} AZ_j^+ &= 0, \\ AZ_j^- &= 1, \\ (j &= 1, 2, \ldots, n). \end{aligned} \tag{19}$$

Based on the above analysis, $AZ^+ = [0, 0]$, $AZ^- = [1, 1]$ in this section.

step5: Replace Z^+ and Z^- with AZ^+ and AZ^- in (15). Calculate the euclidean distance between each resource and AZ^+ according to (15).

step6: Obtain the relative closeness between each resource scheme and the absolute ideal solution AZ^{+} according to the formula (16), and rank all resources in descending order of the relative closeness.

step7: Select the resource p_{sel} with the highest value of the relative closeness for task t_{curr}, and assign t_{curr} to resource p_{sel} for execution.

The process of our algorithm for dynamic multi-workflow scheduling is shown in Algorithm 1.

Algorithm 1 Multi-workflow Scheduling Technology (MT)

Input: The set of given resources $P = \{p_1, p_2, \cdots, p_{\mathsf{P}}\}$, the set of workflows dynamically arrived $GS = \{G_1, G_2, \cdots, G_{\mathsf{M}}\}$
Output: The maximum completion time $Makespan(GS)$ and cost $C(GS)$
while $GS \neq \emptyset$ **do**
  if New workflow G_{new} arrives **then**
    Calculte all tasks' $Rank_n$ of G_{new} and put all tasks in the $init_{queue}$ of G_{new} in the descending order of $Rank_n$
    Add G_{new} into GS
    Return unfinished tasks in *ready_pool* to the *init_queue* of the corresponding workflows for the currently unscheduled completed workflows
  end if
  for $G_i \in GS$ **do**
    Put the task with the highest $Rank_n$ of G_i into *ready_pool*
  end for
  Obtain $Rank_r$ of tasks in $ready_{pool}$
  while *ready_pool* $\neq \emptyset$ **do**
    Select the task t_{curr} task with the highest $Rank_r$ in *ready_pool*
    Sort all resources according to the resource insertion and TOPSIS method
    Select resource p_{sel} with the highest related closeness and delete t_{curr} in *ready_pool*
    Put t_{curr} into the task waiting pool of p_{sel}
    if p_{sel} is idle **then**
      Execute t_{curr} on resource p_{sel}
    else
      Keep t_{curr} in the task waiting pool of p_{sel} and wait for it to be scheduled
    end if
  end while
end while

Table 1 Parameters for experiments

Parameter	Value
$w_{j,k}$	$10us \le w_{j,k} \le 100us$
$c_{i,j}$	$10us \le c_{i,j} \le 100us$
price(p_k)	$0.3\$/h \le \text{price}(p_k) \le 0.7\$/h$
price(p_{comm})	0.1\$/h

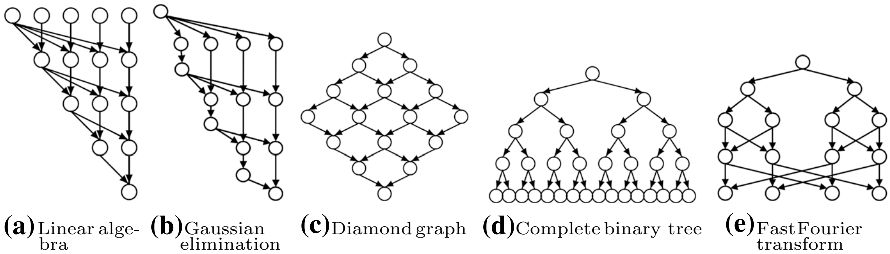

Fig. 5 Five types of workflows

5 Experiments

5.1 Test environments and metrics

The experiment is implemented in the heterogeneous experimental environment (2.11 ghz, 8GB RAM) using java language. The heterogeneous experimental environment consists of several VMs which are with different service unit prices and computing capacities. The parameters for experiments are shown in Table 1. The unit price per unit time of resources is set as $0.3\$/h \le \text{price}(p_k) \le 0.7\$/h$. The unit price of data transmission between resources is set as $\text{price}(p_{\text{comm}}) = 0.1\$/h$. The task execution time is denoted by $10us \le w_{j,k} \le 100us$ which means the different computing capacities of VMs. The Larger $w_{j,k}$ means lower computing capacity for a VM.

In addition, the five types of workflows used in the experiment are shown in Fig. 5. They are linear algebra, Gaussian elimination, Diamond graph, Complete binary tree, and Fast Fourier transform, respectively. To illustrate the number of tasks of each workflow, we introduce a parameter ρ.

(a) Linear algebra: The total number of tasks is $|N| = \rho(\rho + 1)/2$. And the Fig. 5a is the worklfow with $\rho = 5$.

(b) Gaussian elimination: The total number of tasks is $|N| = \frac{\rho^2+\rho-2}{2}$. And the Fig. 5b is the worklfow with $\rho = 5$.

(c) Diamond graph: The total number of tasks is $|N| = \rho^2$. And the Fig. 5c is the worklfow with $\rho = 4$.

(d) Complete binary tree: The total number of tasks is $|N| = 2^\rho - 1$. And the Fig. 5d is the worklfow with $\rho = 5$.

Table 2 Workflow types for experiments

Workflow name	Task number
linear algebra	120
Guassian elimination	119
Diamond graph	121
Complete binary	127
Fast Fourier transform	95

(e) Fast Fourier transform: The total number of tasks is $|N| = 2 \times \rho - 1 + \rho \times log_2\rho$, where $\rho = 2^y$. And the Fig. 5e is the worklfow with $\rho = 4$.

There are more than one entry task in the workflow of Linear algebra, so we add a virtual entry task and all the actual entry tasks are set as the immediate successor tasks of the virtual entry task. There are more than one exit task in the complete binary tree and Fast Fourier transform workflows. Therefore, a virtual exit task should be added and all the actual exit tasks should be set as the immediate predecessor tasks of the virtual exit task. Note that, the data transmission time between the added virtual tasks and the actual tasks is zero.

Furthermore, the number of tasks in linear algebra, Guassian elimination, Diamond graph, Complete binary and Fast Fourier transform workflows are shown in Table 2. The proportion of the five types of workflows is the same under different number of workflows, that is, when the number of multiple workflows is 10, the number of workflows of each type is 2, and when the number of multiple workflows is 20, the number of workflows of each type is 4. Moreover, the workflow parameters are set as $10us \le w_{i,k} \le 100us, 10us \le c_{ij} \le 100us$ in Table 1.

The performance metrics of the algorithm in this paper are the maximum completion time of multi-workflow Makespan(GS) and the total cost of the system C(GS). All of the results for each experiment are average values after 20 times of the dynamic and multiple workflows' execution.

5.2 Experiments results

In this subsection, the effectiveness of MT algorithm is validated. And we show the effectiveness of the algorithm from three aspects: the number of workflows, the arrival time interval of workflow and the number of resources.

5.2.1 Varying number of workflows

The compared algorithms are OWM, FDWS and MPOSH, where MPOSH is the extension of POSH to the problems of multiple workflow by simply adding the function of receiving multiple workflows. This part evaluate the performance of four algorithms by different workflow numbers.

In Fig. 6, the workflow arrival interval is 30, and the number of workflows is set in the range of 10, 20, 30, 40, 50. The number of resources provided is 100.

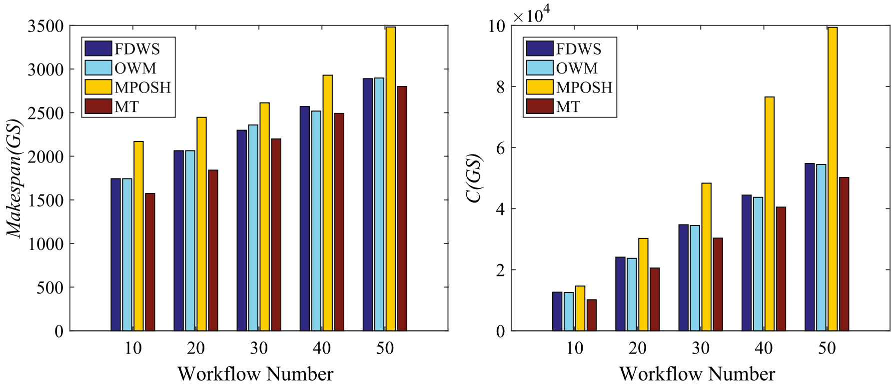

Fig. 6 Makespan(GS) and *C*(GS) values of the different number of workflows

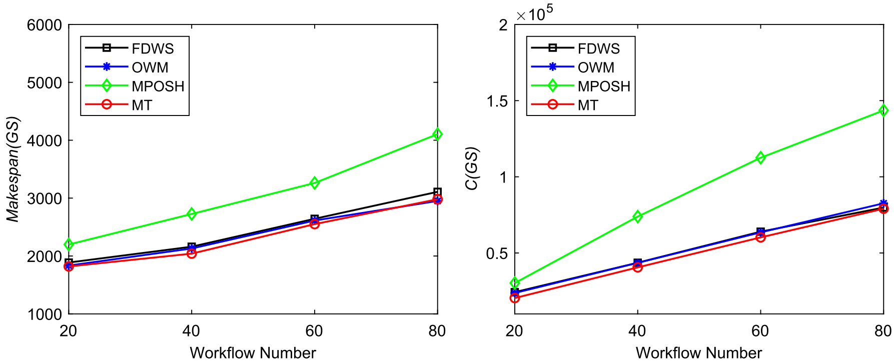

Fig. 7 Makespan(GS) and *C*(GS) values of the different number of workflows

According to Fig. 6, the makespan of four algorithms are increasing with the increase of the workflow number. The reason is that the set of given resources is fixed, and the increase of the amount of tasks to be processed will inevitably lead to the increase of time consumption and cost. In addition, it can be seen that the makespan and cost of MT algorithm are lower than those of the other three algorithms under different workflow numbers. This is mainly because the MT algorithm considers the heterogeneity of VMs for sorting tasks and adopts the TOPSIS method to rank the resources for executing tasks, which can further reduce the execution time and cost of the workflow. Furthermore, to fully demonstrate the benefits of the proposed algorithm, we set the workflow arrival interval as 10 and the number of resources as 100. The number of workflows is set in the range of 20, 40, 60, 80 which is used to evaluated the makespan and cost of four algorithms for workflows with different workflow numbers in Fig. 7. From Fig. 7, the makespan of four algorithms are increasing with the increase of the workflow number. In addition, it can be seen that the makespan and cost of MT algorithm are lower than those of the

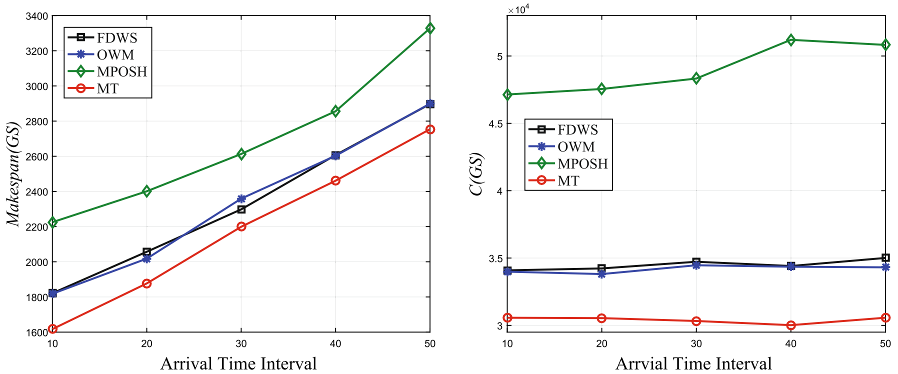

Fig. 8 Makespan(GS) and *C*(GS) values of the different arrival interval of workflows

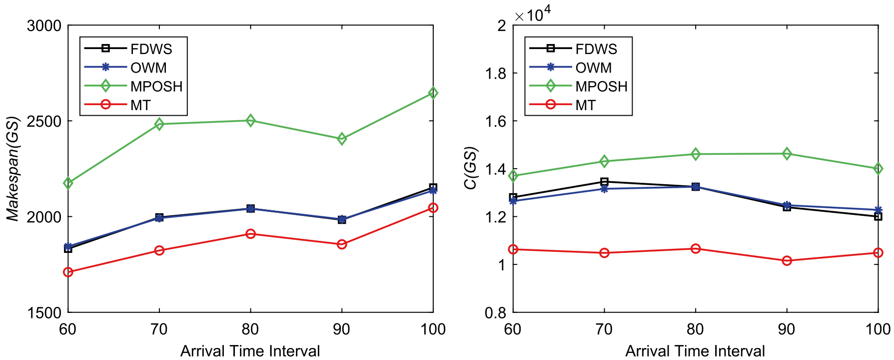

Fig. 9 Makespan(GS) and *C*(GS) values of the different arrival interval of workflows

other three algorithms under different workflow numbers. Hence, the experimental results demonstrate that MT outperforms three state-of-the-art algorithms in terms of makespan and cost with different workflow numbers.

5.2.2 Varying the arrival time interval of workflow

In this subsection, the performance of FDWS algorithm, OWM algorithm, MPOSH algorithm and MT algorithm are compared from the perspectives of maximum completion time and total system cost of multiple workflows by different arrival time interval of multiple workflows. The number of workflows in this part is 30, and the number of resources is 100. The experimental results are shown in Figs. 8 and 9.

In Fig. 8, the arrival time interval of workflow is set to 10, 20, 30, 40, 50 respectively. Figure 8 indicates that under the same number of workflows, the makespan of the four algorithms gradually increases with the increase of the arrival time interval of the workflows. Moreover, compared with the other three algorithms, the makespan and cost of multiple workflows obtained by MT algorithm are the smallest. And

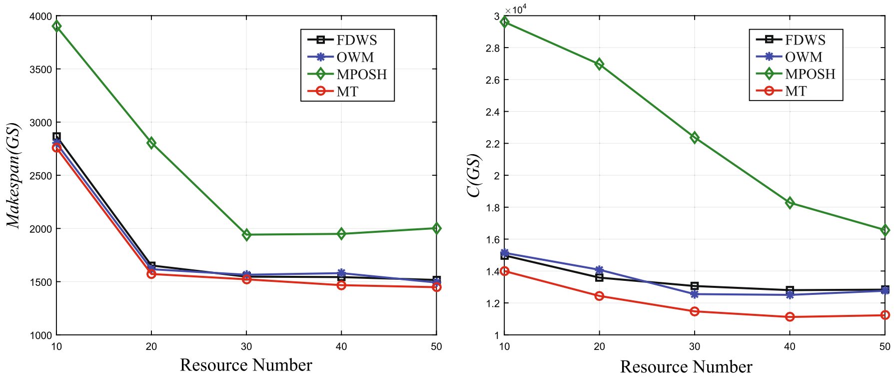

Fig. 10 Makespan(GS) and *C*(GS) values of the different number of resources

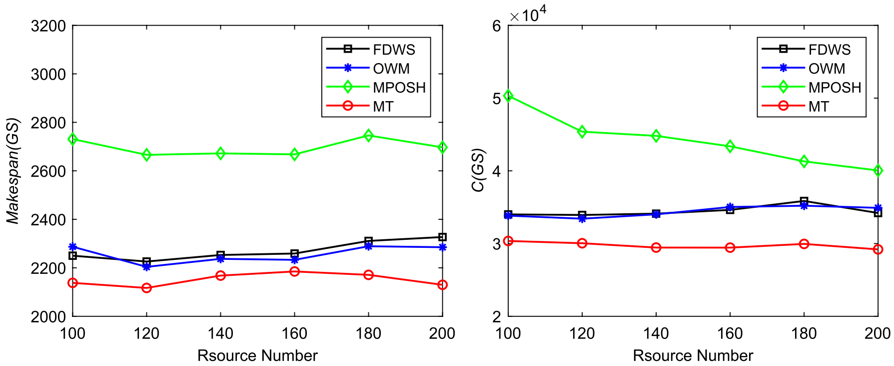

Fig. 11 Makespan(GS) and *C*(GS) values of the different number of resources

the performance of MPOSH algorithm is worse than FDWS and OWM algorithms. Furthermore, the arrival time interval of workflow is set to 60, 70, 80, 90 and 100 in Fig. 9. From Fig. 9, the makespan of all algorithms are increasing with the increase of the arrival time interval. In addition, it can be seen that the makespan and cost of MT algorithm are lower than those of the other three algorithms under different arrival time interval. Hence, We can observe that the performance of MT is better compared with the FDWS, OWN and MPOSH in these experiments.

5.2.3 Varying number of resources

We evaluate the performance of four algorithms with different number of resources in this part. The experimental results of the four algorithms when the number of resources changes as shown in Figs. 10 and 11.

In Fig. 10, the number of resources is set to 10, 20, 30, 40 and 50, respectively. The number of workflows set in this part is 10, and the arrival time interval of workflow is 10. As depicted in Fig. 10, with the change of the number of resources, the

MT algorithm can also achieve the best performance in the maespan and the cost compared with the other three algorithms. It shows that the MT algorithm has a good robustness when there has a change in the number of resources provided. This is mainly because that MT algorithm adopts absolute ideal solution of 0-1 type instead of relative ideal solution to ensure the uniqueness of evaluation criteria and avoid the reverse order of the same two resources when it uses the TOPSIS method to select resources for tasks. To further demonstrate the effectiveness of MT, the number of workflows is set as 30, and the arrival time interval of workflow is 30. The number of resources is set to 100, 120, 140, 160, 180 and 200 in the experiments, respectively. The experimental results of the four algorithms when the number of resources changes as shown in Fig. 11. It can be seen from Fig. 11 that the makespan and cost of MT algorithm are lower than those of the other three algorithms under different resource numbers. Hence, We can observe that the performance of MT is better compared with the FDWS, OWN and MPOSH in these experiments.

6 Conclusion

In this paper, we focus on the optimization of maximum completion time and total cost for the dynamic multi-workflow. Firstly, the system model and framework of dynamic multi workflow scheduling are proposed. Secondly, the optimization algorithm of dynamic multi workflow scheduling based on TOPSIS is designed, and the task priority calculation method, selection process and resource selection based on TOPSIS in heterogeneous computing environment are given. Finally, experiments hava been carried out in five types of real workflows, which fully prove the superiority and effectiveness of the scheduling algorithm proposed in this paper in reducing the maximum completion time and saving the total cost of the multiple workflows. In the future, we intend to extend MT to other workflow scheduling problems such as energy-aware and privacy-aware workflows scheduling in clouds.

Acknowledgements All codes and data are published on GitHub [21]. There is no conflict of interest in this paper.

References

1. Xia Y (2015) Cloud control systems. IEEE/CAA J Autom Sinica 2(2):134–142
2. Xia Y, Yan C, Wang X (2019) Intelligent transportation cyber-physical cloud control systems. Acta Autom Sinica 45(1):132–142
3. Atmaca T, Begin T, Brandwajn A, Casteltaleb H (2016) Performance evaluation of cloud computing centers with general arrivals and service. IEEE Trans Parallel Distrib Syst 27(8):2341–2348
4. Xie G, Chen Y, Li R, Li K (2018) Hardware cost design optimization for functional safety-critical parallel applications on heterogeneous distributed embedded systems. IEEE Trans Industr Inf 14(6):2418–2431
5. Yassir S, Mostapha Z, Claude T (2017) Workflow scheduling issues and techniques in cloud computing: a systematic literature review. In: Proceedings of the International Conference of Cloud Computing Technologies and Applications, pp. 241– 263 . Springer
6. Bardsiri AK, Hashemi SM (2012) A review of workflow scheduling in cloud computing environment. Int J Comput Sci Manag Res 1(3):348–351

7. Topcuoglu H, Hariri S, Wu M-y (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
8. Braun TD, Siegel HJ, Beck N, Bölöni LL, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen D et al (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Comput 61(6):810–837
9. Sakellariou R, Zhao H, Tsiakkouri E, Dikaiakos MD (2007) Scheduling workflows with budget constraints, 189–202
10. Assari A, Mahesh T, Assari E (2012) Role of public participation in sustainability of historical city: usage of topsis method. Indian J Sci Technol 5(3):2289–2294
11. Lin C, Lu S (2011) Scheduling scientific workflows elastically for cloud computing. In: Proceedings of the 4th IEEE International Conference on Cloud Computing, pp. 746– 747 . IEEE
12. Rodriguez MA, Buyya R (2014) Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds. IEEE Trans Cloud Comput 2(2):222–235
13. Chen Z-G, Zhan Z-H, Li H-H, Du K-J, Zhong J-H, Foo YW, Li Y, Zhang J (2015) Deadline constrained cloud computing resources scheduling through an ant colony system approach. In: 2015 International Conference on Cloud Computing Research and Innovation, pp. 112– 119 . IEEE
14. Su S, Li J, Huang Q, Huang X, Shuang K, Wang J (2013) Cost-efficient task scheduling for executing large programs in the cloud. Parallel Comput 39(4–5):177–188
15. Li H, Wang B, Yuan Y, Zhou M, Fang Y, Xia Y (2021) Scoring and dynamic hierarchy-based nsga-ii for multiobjective workflow scheduling in the cloud. IEEE Trans Autom Sci Eng. https://doi.org/10.1109/TASE.2021.3054501
16. Chen Z-G, Zhan Z-H, Lin Y, Gong Y-J, Gu T-L, Zhao F, Yuan H-Q, Zhang J (2019) Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach. IEEE Trans Cybern 49(8):2912–2926
17. Zhu Z, Zhang G, Li M, Liu X (2016) Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans Parallel Distrib Syst 27(5):1344–1357
18. Garg R, Singh AK (2011) Multi-objective workflow grid scheduling based on discrete particle swarm optimization. In: International Conference on Swarm, Evolutionary, and Memetic Computing, pp. 183– 190 . Springer
19. Hsu C-C, Huang K-C, Wang F-J (2011) Online scheduling of workflow applications in grid environments. Futur Gener Comput Syst 27(6):860–870
20. Lai Y-J, Liu T-Y, Hwang C-L (1994) Topsis for modm. Eur J Oper Res 76(3):486–500
21. Zhan, Y (2022) https://github.com/Ray-ZHAN/workflow-scheduling. [Online; accessed 06-Jun.-2022]

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.