Journal of King Saud University – Computer and Information Sciences xxx (xxxx) xxx

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

TOPSIS inspired cost-efficient concurrent workflow scheduling algorithm in cloud

K. Kalyan Chakravarthi [a,*], L. Shyamala [a], V. Vaidehi [b]

[a] School of Computer Science and Engineering, VIT Chennai, India
[b] Mother Teresa Women's University, Tamil Nadu, India

A R T I C L E I N F O

Article history:
Received 11 December 2019
Revised 21 February 2020
Accepted 26 February 2020
Available online xxxx

Keywords:
TOPSIS
Quality of Service
Deadline
Scheduling
Concurrent Workflows

A B S T R A C T

Scheduling is a decision-making mechanism that enables the sharing of resources among several activities by determining their execution order on the set of available resources. In distributed systems, it is a great challenge to schedule multiple workflows submitted at different times. In particular, concurrent workflow scheduling with time constraints makes the problem more complex in the cloud due to the dynamics of the cloud such as elasticity, non-homogeneous resource types, various pricing schemes, and virtualization. A well-managed deadline workflow scheduling is required to improve end-user satisfaction and system performance. In the meantime, the intrinsic uncertainty in the cloud increases the difficulties of scheduling problems. Therefore, it is a great challenge to improve system performance and optimize several scheduling criteria simultaneously. To address the above issues, a novel concurrent workflow scheduling method for heterogeneous distributed environments based on the new Multi-Criteria Decision Making (MCDM) method i.e., TOPSIS (Technique of Order Preference by Similarity to Ideal Solution) is presented. A weighted sum of execution time, cost and communication time are used to find out the optimal resource among the existing resources as per the workflow task requirements. The proposed method minimizes the makespan and execution cost of the workflow and improves the resource efficiency under uncertain environment. The performance of the proposed work is compared with the state-of-the-art algorithms such as Cloud-based Workflow Scheduling Algorithm (CWSA), Earliest Finish Time-Maximum Effective Reduction (EFT-MER) and Heterogeneous Earliest-Finish-Time (HEFT) algorithms based on deadline constraint and resource utilization. Our experimental results demonstrate that the proposed T-CCWSA outperforms current state-of-the-art heuristics with the criteria of achieving the deadline constraint, minimizing the cost of execution and resource efficiency.

1. Introduction

Complex scientific applications like Bio-informatics, Earth Science, Astronomy and disaster modeling can be represented naturally in the form of workflows (Arabnejad et al., 2019; Partheeban and Kavitha, 2018). One of the benefits of workflow representation is that the workflow can be reusable, reproducible and even traceable through other workflows (Guo et al., 2018; Iyenghar and Pulvermueller, 2018). In the meantime, computing systems where catastrophe can occur will become useless, if the completion of workflow execution takes more than some specified time. These workflow-based applications are highly demanding and challenging for processing huge amounts of data in real-time workflow tasks with the desired cost reduction of computing resources (Ghafouri et al., 2018).

Workflow scheduling is a process of mapping the workflow tasks to the computing resources. The relationship among workflow tasks is multiple inter-dependent due to complex operations. Therefore, the inter-relationship between parent-child and the data/control dependencies are represented by the Directed Acyclic Graph (DAG). Each workflow task has various execution times, priorities and time limits associated with the other workflow (Emmanuel et al., 2018; Zhou et al., 2018). Efficient workflow scheduling is essential to achieve user Quality of Service (QoS) requirements, e.g. minimizing execution time and at the same time maximizing the system performance, e.g. use of resources.

* Corresponding author.
E-mail address: kalyan.koneti@gmail.com (K.K. Chakravarthi).

Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2020.02.006
1319-1578/© 2020 The Authors. Production and hosting by Elsevier B.V. on behalf of King Saud University.

Moreover, in order to achieve good efficiency, various workflow applications require different scheduling strategies in distributed systems. Most of the literature investigated the homogeneous computing system, while some studies centered on heterogeneous systems each with different QoS constraints. The existing methods tend to be process-oriented and/or data-oriented rather than resource-oriented, and they lack efficient task to resource mapping strategies (Aziz et al., 2015). In addition, in traditional scheduling, parameters like task communication time and computation time have been considered to be deterministic (Khorsand et al., 2016; Wang et al., 2018; Zhou et al., 2018; Su et al., 2015). In fact, in real-world situations, various factors involved in the scheduling problems are often imprecise or uncertain in nature (Zavadskas and Podvezko, 2016; Pan and Deng, 2018; Jiang and Hu, 2018; Sun and Deng, 2019). Especially, human-defined factors are involved in the scheduling problems. Until now, researchers have drawn considerable attention to the problems of modeling and handling of uncertain information (Yager, 2018; He and Jiang, 2018; Kang et al., 2019; Yang et al., 2018; Deng, 2018; Zavadskas et al., 2017).

With the vast proliferation of cloud resources with varying configurations, it is challenging to choose an optimal resource to satisfy and fulfill the business strategies and satisfy the users' QoS requirements, with objectives that are often contradictory with one other (Cusumano, 2010; Assunção et al., 2010; Jatoth et al., 2017; Martino et al., 2016; Hajji and Mezni, 2017; Capuano et al., 2018; Capuano et al., 2017; Carrasco et al., 2015). The most suitable resource should be sought, considering multiple incompatible qualitative and quantitative criteria. It is quite significant to consider how to optimize workflow scheduling while satisfying the various QoS requirements of users. Therefore, the selection of cloud resources can be considered as a MCDM problem (Abdel-Basset et al., 2018). MCDM's objective is to identify the best alternative among other alternatives in the presence of several adverse decision criteria. In scheduling problem, MCDM's goal is to evaluate and rank alternatives (VMs) based on the user's QoS constraints.

Of late, MCDM methods provided solutions to several real-world problems. MCDM is an operations research sub-discipline that explicitly identifies the best alternative from the pool of potential alternatives (Chen et al., 2008; Yue, 2011) by analyzing several attributes or criteria that may be concrete or vague. From the last few decades, the MCDM techniques have been built in the subjective order of preference to identify, classify, and choose alternatives (Behzadian et al., 2012). MCDM has been widely used in many fields (Abdel-Basset et al., 2018; Abdel-Basset et al., 2019), and in various MCDM studies, the Technology for Order Preference by Similarity to an Ideal Solution (TOPSIS) method (Hwang and Yoon, 1981) has been successfully applied in different fields (Shidpour et al., 2013; Chang et al., 2018). Originally, TOPSIS was developed by Hwang and Yoon (Hwang and Yoon, 1981) to find the best alternative, identified as the one with the shortest distance from the Positive Ideal Solution (PIS) and the longest distance from the Negative Ideal Solution (NIS) (Olson, 2004; Kao, 2010; Wang and Luo, 2010; Lotfi et al., 2013; Sarraf et al., 2013). The TOPSIS method is more efficient, robust and simpler than other MCDM models. Furthermore, the number of criteria and alternatives in TOPSIS is not limited. The TOPSIS method has a generous impact on real-world decision-making problems and works well for various applications (Bulgurcu, 2012; Mir et al., 2016). Still, less attention is paid to the use of MCDM by researchers in cloud computing. Our proposal using TOPSIS addresses these issues in the selection process of resources and optimizes workflow scheduling in the cloud.

This paper's significant contributions are summarized below.

- Proposed a Multi-Criteria Decision-Making model for ranking the cloud resources based on QoS parameters.
- A novel algorithm named T-CCWSA for scheduling dynamic workflows.
- Implemented the TOPSIS algorithm with T-CCWSA to achieve optimal solutions to perform workflow scheduling.

The rest of the work is organized as follows. Section 2 provides a precise study of literature on workflow scheduling and highlights the major works and proposals. Section 3 gives an overview of problem formulation. In continuation, Section 4 discusses the generalized mechanism of TOPSIS and the detailed description of the proposed solution to the workflow scheduling problem shall be discussed in Section 5. Section 6 delivers the evaluation report of the arrived results and its comparison charts. Towards the end of this paper, in Section 7, the possible scope of future work in the workflow scheduling is suggested.

2. Related work

Recently, workflow scheduling in heterogeneous distributed environments has received significant interest, and various studies presented numerous workflow scheduling approaches to achieve near-optimal solutions. Workflow scheduling can be classified as either single or multi workflow scheduling. Single workflow scheduling approaches can be roughly divided into QoS constraint scheduling and Best-effort scheduling (Yu et al., 2008). The Best-effort scheduling (Arabnejad and Barbosa, 2014; Daoud and Kharma, 2008; Zhou et al., 2016) aims to minimize the schedule length while ignoring the QoS constraints of various users. The QoS constraint scheduling (Abrishami et al., 2012; Arabnejad and Barbosa, 2014; Azad and Navimipour, 2017; Wu et al., 2016; Zheng and Sakellariou, 2013) tries to improve the scheduling performance under QoS constraints, such as minimizing time under budget constraint or minimizing cost under deadline constraint.

Multiple workflow scheduling can be divided into either offline or online scheduling. Offline scheduling does not deal with workflows arrived after the schedule is generated. In the literature, many algorithms for offline scheduling have been proposed. Zhao and Sakellariou (Zhao and Sakellariou, 2006) presented two fairness-based strategies: finish time fairness policy and current time fairness policy. In both policies, fairness is based on the principle of "slowdown", it is the ratio of expected execution time when a workflow is scheduled alone and the expected execution time when all workflows are scheduled together. The difference between the two policies is that the current time fairness policy determines the slowdown value for all the workflows, whereas the finish time fairness policy estimates the slowdown value for the selected workflow only. However, these two algorithms are only involved in offline scheduling when all the workflows are known in advance.

Xu et al. (2017) proposed an Expansion Slot Backfill (ESB) algorithm for multiple workflow scheduling under deadline constraints. This algorithm's objective is to minimize the number of idle time slots and maximize resource utilization. It schedules all workflows using a sequential strategy based on the priority order. For each workflow, priority is computed based on the resource utilization or the relative urgency degree. In turn, every new task will try to backfill the earliest time slot. If the earliest time slot is not sufficient to backfill, it is extended. Overall, an ESB algorithm makes better use of small-time slots which fail to satisfy new task backfill requirements in comparison with existing approaches.

Online scheduling allows users to submit workflows at any time. A few algorithms were proposed for scheduling online workflows.

Finish Time Fairness algorithm has been modified by Tian et al. (2012) described in Zhao and Sakellariou (2006) and proposed a dynamic E-fairness scheduling algorithm. For each newly arrived workflow, the E-fairness algorithm uses the slowdown value and decides which workflow should be scheduled next. Yu and Shi (2008) proposed the RANK_HYBD algorithm which addresses workflows submitted at different times by various users. This algorithm schedule the lowest rank workflow to minimize the workflow waiting time. But this approach does not achieve high fairness. For example, if the ranks of tasks in subsequent workflows are less than that of unscheduled tasks in previously arrived workflows, the remaining tasks in the prior workflows will not be scheduled.

The Fairness Dynamic Workflow Scheduling (FDWS) algorithm presented in Arabnejad and Barbosa (2012), addressed both the unfairness that occurs in RANK_HYBD (Yu and Shi, 2008) and the postponement of the selection of task that occurs with Online Workflow Management (OWM) (Hsu et al., 2011). The OWM and RANK_HYBD algorithms have been proposed to minimize the average completion time of all workflows. On the other hand, the FDWS algorithm aims to reduce the waiting and execution time of each workflow. The FDWS algorithm exceeds the RANK_HYBD and the OWM algorithms in terms of average makespan, win (%) and, Scheduling Length Ratio (SLR).

Arabnejad et al. (2014) surveyed both offline and online scheduling of concurrent workflows and proposed a modified version of max-min and min-min (Maheswaran et al., 1999) algorithms. The authors compared the FDWS algorithm with OWM, RANK_HYBD, modified max-min, and min-min algorithms, and found that the FDWS performs best. The above online scheduling algorithms address only fairness in the sharing of resources among workflows and minimize the makespan. But in the pay-as-you-go model, users use resources and services and pay only for what they use. Therefore, from the user's point of view, time and cost are the two most important factors. The online strategy outlined in Arabnejad and Barbosa (2014) acknowledges cost and time but aims to minimize each workflow's turnaround time within the user-defined budget.

Arabnejad and Barbosa (2015) and Arabnejad and Barbosa (2017) presented a Multi Workflow Deadline-Budget Scheduling (MWDBS) algorithm for concurrent workflows under deadline and budget constraints. The MWDBS algorithm has a higher success rate than other algorithms, especially for highly concurrent scenarios, i.e. those with lower arrival intervals. However, this approach does have drawbacks under the inconsistent model, because it selects each task based on the upward rank (Topcuoglu et al., 2002) ($rank_u$) priority without considering the cost of execution. Therefore, an acceptable trade-off is not achieved between the users' deadline and budget constraints.

Compared to current scheduling strategies, where the full schedule is generated when a workflow arrives, the proposed approach continuously generates a new schedule for each workflow task during its actual execution time to mitigate the impact of cloud uncertainty on scheduling performance. This work has also attracted attention to a method called TOPSIS based on MCDS. Workflow task mapping is carried out by the TOPSIS algorithm which maps workflow tasks to cloud resources by considering multiple substantial factors. The involvement of the TOPSIS algorithm supports multiple objective functions to optimize scheduling performance.

3. Problem formulation

In the cloud, workflows can be modeled as $W = \{w_1, w_2, w_3 \ldots, w_n\}$. A workflow $w_i \in W$, can be modeled as $w_i = \left\{G_i, QoS_i, T_a^i\right\}$, where G_i, QoS_i, and T_a^i represent workflow's structure, quality of service constraints and arrival time respectively. Workflow's structure G_i of workflow w_i can be further described as a directed acyclic graph (DAG) $G_i = (T_i, E_i)$, where $T_i = \{t_1^i, t_2^i, \ldots, t_N^i\}$ is a set of vertices and N represents the count of workflow tasks; the vertex t_j^i denotes the j^{th} task in a workflow w_i. Also, $E_i \subseteq T_i \times T_i$ denotes the directed edges among tasks. An edge $e_{k,j}^i \in E_i$ of the form $\left(t_k^i, t_j^i\right)$ exists if there is a precedence constraint between the tasks t_k^i and t_j^i, where t_k^i is an immediate predecessor of task t_j^i and the task t_j^i is an immediate successor of task t_k^i. The $pred\left(t_j^i\right)$ denotes the set of tasks consisting of all t_j^i's immediate predecessors, and $succ\left(t_j^i\right)$ represents the set of tasks consisting of all t_j^i's immediate successors. For a certain workflow $w_i \in W$ quality of constraints can be modeled as $QoS_i = \{con^i, con_{val}^i\}$, where con^i and con_{val}^i represent user-specified constraints such as deadline or budget and user-specified constraint value for deadline or budget.

The cloud platform provides an infinite number of VMs with various configurations (Zhu et al., 2016; Calheiros and Buyya, 2014). Let $VT = \{vt_1, vt_2, vt_3 \ldots, vt_m\}$ represent m types of virtual machines available in the cloud. The parameter $vm_k^{vt_u}$ represents the k^{th} VM of type vt_u. The price of the virtual machine denoted as $price(vm_k^{vt_u})$ is the cost per unit interval of time. It is worth noting that virtual machines can be acquired and terminated at any point in time. Also, virtual machines are charged per unit interval of time, and any partial use of the unit of time will be charged for the whole period.

In multiple workflow scheduling, the symbol $ET_{j,k}^i$ denotes the execution time of a task t_j^i on VM $vm_k^{vt_u}$. Besides, the symbols $EST_{j,k}^i$ and $EFT_{j,k}^i$ denotes the earliest start time and earliest finish time of a task t_j^i on VM $vm_k^{vt_u}$.The earliest time at which a task t_j^i can begin its execution on VM $vm_k^{vt_u}$ is known as the earliest start time and calculated as follows.

$$EST_j^i = \begin{cases} T_a^i, & \text{if pred}\left(t_j^i\right) = NULL \\ \max\limits_{t_p^i \in \text{pred}\left(t_j^i\right)} \left\{EST_p^i + MET_p^i + MTT_{p,j}^i\right\}, & \text{otherwise} \end{cases} \tag{1}$$

where $MTT_{p,j}^i$ denotes the minimum data communication time from predecessor task t_p^i to current task t_j^i and MET_j^i represent the minimum execution time of a task t_j^i on VM $vm_k^{vt_u} \in VT$ that have minimum execution time among all VM types in the cloud and computed as

$$MET_j^i = \min_{vm_k^{vt_u} \in VT} \left\{ET_{j,k}^i\right\} \tag{2}$$

Apparently, the estimated finish time, $EFT_{j,k}^i$ can be computed as

$$EFT_{j,k}^i = EST_{j,k}^i + MET_{j,k}^i \tag{3}$$

In an arbitrary workflow scheduling, the latest finish time LFT_j^i of task t_j^i is the period before which task completes its computation, such that finish time of workflow w_i is less than the user-specified deadline, d_i. It is defined as

$$LFT_j^i = \begin{cases} d_i, & \text{if succ}\left(t_j^i\right) = NULL \\ \min\limits_{t_p^i \in \text{succ}\left(t_j^i\right)} \left\{LFT_p^i - MET_p^i - MTT_{p,j}^i\right\}, & \text{otherwise} \end{cases} \tag{4}$$

Due to precedence constraints in a workflow, the task can not be executed until it gathers all of the data from its immediate predecessors, we have the following constraint.

$$FT_{p,k}^i + DT_{p,j}^i \leqslant ST_{j,k}^i \quad \forall e_{p,j}^i \in E_i \tag{5}$$

where $FT^i_{p,k}$ denotes task t^i_p's finish time on VM $vm^{vt_u}_k$ and $DT^i_{p,j}$ denotes the data transfer time between task t^i_p and t^i_j. In workflow scheduling environments, finish time FT^i of workflow w_i is the maximal finish time of all its tasks and defined as

$$FT^i = \max_{t^i_j \in (w_i)} \left\{ FT^i_{j,k} \right\} \tag{6}$$

The cost $c^i_{j,k}$ for executing task t^i_j on VM $vm^{vt_u}_k$ is calculated as

$$c^i_{j,k} = price(vm^{vt_u}_k) * \left\{ \frac{ET^i_{j,k}}{N_t} \right\} \tag{7}$$

where $price(vm^{vt_u}_k)$ is the cost of the VM $vm^{vt_u}_k$ per one interval of time and N_t is the unit of time interval.

To ensure the deadline of a workflow, all the workflow tasks in w_i must finish their execution before its deadline d_i. Consequently, this brings with it another constraint

$$FT^i \leqslant d_i, \quad \forall w_i \in W \tag{8}$$

Subjecting to aforementioned constraints in (5) and (8), the primary goal of optimization is to minimize the Total Execution Cost (TEC) of completing workflow set W, which can be computed as

$$\text{Minimize } TEC = \sum_{k=1}^{|VM|} price(vm^{vt_u}_k) \cdot p_k \tag{9}$$

where |VM| represents the number of acquired VMs and p_k denotes the number of time units of leased VM $vm^{vt_u}_k$.

Resource utilization is also an important metric for cloud service providers to evaluate the performance of a cloud platform. Therefore, we are also focusing on maximizing the average resource utilization of VMs, that can be defined as

$$\text{Maximize} \sum_{k=1}^{|VM|} (wt_k) / \sum_{k=1}^{|VM|} (at_k) \tag{10}$$

where wt_k and at_k denote VM's $vm^{vt_u}_k$ working time and active time (idle time + working time).

4. TOPSIS approach

This section introduces the basic concept of Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). The TOPSIS algorithm is presented in Algorithm 1 and steps are given below.

Step a Constitute a Decision Matrix (D) of size p x q, where p represents the number of alternatives and q represents the number of criteria respectively. Alternatives are denoted by Virtual Machines (VMs) and criteria are represented by objective functions as shown in Table 1.

Step b Transform the dimensional attributes into non-dimensional attributes by comparing against each criterion. The decision matrix in Table 1 is normalized using Eq. (1)

$$DM_{ij} = \frac{X_{ij}}{\sqrt{\sum_{i=1}^{p} X^2_{ij}}} \tag{11}$$

where i={1,2,3, ..., p}, j={1,2,3, ..., q} and X_{ij} denotes i^{th} alternative and j^{th} criteria in matrix D.

Step c For each criterion, weight values are given in accordance with the relevance of the criteria in the scheduling process. A weighted normalized decision matrix is generated by multiplying each element (D_{ij}) in normalized D with the weight values corresponding to each criterion.

Table 1
Decision Matrix.

	ET	DT	C
VM_1	ET_{11}	DT_{12}	C_{13}
VM_2	ET_{21}	DT_{22}	C_{33}
VM_3	ET_{31}	DT_{32}	C_{33}
–	–	–	–
–	–	–	–
VM_p	ET_{p1}	DT_{p2}	C_{p3}

$v_{ij} = w_j * D_{ij}$ where i={1,2,3, ..., p}, j={1,2,3, ..., q} and w_j denotes the weight value of the j^{th} objective.

Step d Positive Ideal Solution S^+ identified with the value of criteria having a positive impact and the Negative Ideal Solution S^- associated with the value of criteria having a negative impact on the solution are given by

$$S^+ = \{V^+_1, V^+_2, V^+_3, \ldots, V^+_q\}$$

$$S^- = \{V^-_1, V^-_2, V^-_3, \ldots, V^-_q\}$$

In this context, ET, DT, and C are seen as criteria that have a positive impact and must be minimized

Step e Calculate separation measures. The separation of each alternative from Positive Ideal Solution S^+ and Negative Ideal Solution S^- are computed as,

$$S^* = \sqrt{\sum_{j=0}^{3} \left(V_{ij} - V^+_j \right)^2}$$

$$S' = \sqrt{\sum_{j=0}^{3} \left(V_{ij} - V^-_j \right)^2}$$

where S^* and S' denotes the shortest distance from Positive Ideal Solution S^+ and the longest distance from Negative Ideal Solution S^- to the alternative respectively and j represent the number of criteria.

Step f Compute Relative Closeness (RC) to the ideal solution with respect to S^+ and defined as,

$$RC_i = \frac{S'}{S' + S^*}$$

Step g Rank RC_i according to the preference

Algorithm 1: TOPSIS

```
Input : Task, ET, DT, and C
Output : Ranked Computing Resources
Set criteria as ET, DT, and C
Compute Decision Matrix
$D[ET] \leftarrow SizeofTask/MipsOfVM$
$D[DT] \leftarrow FileSizeOfTask/BandwidthOfVM$
$D[C] \leftarrow ProcessingCostPerUnitTime * ExectionTimeOfTask$
Normalize the Decision Matrix
Compute Weighted Normalized Decision Matrix
Calculate $S^+$ and $S^-$
$S^+$ ←set of benefit attributes i.e., more is better
$S^-$ ←set of negative attributes i.e., less is better
Determine the separation of measures from $S^+$ and $S^-$ for each
  alternative
Compute Relative Closeness (RC)
Rank the VMs
```

5. TOPSIS inspired cost-efficient concurrent workflow scheduling (T-CCWSA) algorithm

Workflow scheduling is widely known to be an NP-complete problem, and finding the optimal schedule within an acceptable time is not feasible. This paper proposes a T-CCWSA algorithm, which allows resource provisioning and scheduling decisions to satisfy each workflow's deadline while minimizing the execution cost. The algorithm generally maintains a resources pool that is scaled in and out according to the current task requirements that are ready for execution. Its main objective is to make efficient use of these resources as a cost-controlling mechanism without compromising the workflow deadline.

In traditional scheduling, when a new workflow arrives, all of the workflow tasks are immediately mapped to the virtual machines. Unlike others, the proposed method adds the waiting tasks to the taskPool and only ready tasks are mapped to the VMs. The T-CCWSA algorithm consists of five phases: Task Merging, Workflow Pre-Processing, WQ Task Processor, Task Scheduler, Task Monitor.

5.1. Task merging

To reduce the runtime overhead of the algorithm, the workflow is pre-processed to merge pipeline tasks (Figs. 1 and 2) into a single task. It helps to save the data transfer time to the next pipeline stage and also accelerates the generation of dynamic scheduling/provisioning plan. When the user submits a new workflow, Tasks Merge Algorithm merges the pipeline tasks into a single task.

Intuitively, task merge can prevent data communication time from delaying workflow tasks' start time. Here we illustrate how to merge tasks' predecessors to minimize tasks' starting time.

Scenario 1: A scenario which focuses on minimizing the task's starting time with only one predecessor

Let us assume two tasks t_j^i and t_k^i are sequence tasks and t_k^i is the successor of t_j^i. Since task t_k^i can not start its execution till it receives the dataset from the task t_j^i , the start time of the task t_k^i should satisfy the condition $ST_{k,r}^i \geqslant ST_{j,s}^i + ET_{j,s}^i + DT_{s,k}^i$. On merging the tasks t_j^i and t_k^i , the communication time between tasks t_j^i and t_k^i is zero. The sequence tasks merging steps are given below.

Step a: For a task t_j^i, if it has only one successor t_k^i and t_k^i has only one parent t_j^i then replace t_j^i and t_k^i with the task t_{j+k}^i

Step b: Set t_{j+k}^i as the parent of t_k^i's children

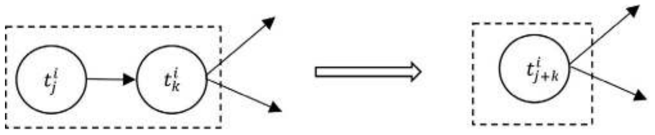


Fig. 1. The process of sequence tasks merging.

Fig. 1 depicts the sequence tasks merging process. The advantages of sequence tasks merging have twofold: 1. All the merged tasks are sequence tasks without impacting the tasks' precedence rules. 2. Reduces task monetary cost as we do not need to transfer the dataset.

Scenario 1 is illustrated with an example in Fig. 1 Let us assume the task t_k^i has only one predecessor task t_j^i as shown in Fig. 1, and task t_j^i has been scheduled to VM $vm_s^{vt_u}$ with finish time i.e., $ST_{j,s}^i + ET_{j,s}^i$ = 40s. Besides, the data communication time is assumed to be $DT_{s,k}^i = 15s$. According to Scenario 1, we merge two tasks t_j^i and t_k^i . It helps to avoid the communication time between tasks t_j^i and t_k^i when they run on different resources and also reduces the run time overhead.

Scenario 2: A scenario which focuses on minimizing the task's starting time with multiple predecessor tasks

Let us assume that task t_j^i contains multiple predecessor tasks, and $pred\left(t_j^i\right) = \{t_1^i, t_2^i, \ldots, t_k^i\}$. Let $\max ET_{sub}\left(t_j^i\right)$ as the maximum finish time among the predecessors of t_j^i, denoted by $\max ET_{sub}\left(t_j^i\right) = \max_{t_l^i \in \mathrm{pred}\left(t_j^i\right)} \{ET(t_l^i)\}$. If the finish time of two merged tasks does not exceed the maximum finish time $\max ET_{sub}$, then the tasks could be merged together. Multiple predecessor tasks merging implementation steps are outlined below.

Step a: For a task t_j^i find out all predecessor tasks $grouppred\left(t_j^i\right) = \{t_1^i, t_2^i, \ldots, t_k^i\}$.

Step b: Sort the tasks according to their finish time in ascending order

Step c: Take two tasks which have the minimum finish time. If $ET(t_1^i + t_2^i) \leqslant ET(t_3^i)$ then merge the two tasks as a new taskt_{1+2}^i

Step d: Update the predecessors, successors, Finish time and resort the tasks according to new finish time

Step e: Repeat steps b to d until the predecessor group is empty

Fig. 2 depicts the procedure of merging predecessor tasks. The advantages of predecessor tasks merging have twofold: a. It does not affect the successors' execution time b. The execution cost of tasks can be reduced.

Scenario 2 is illustrated with an example in Fig. 2. Assume that task t_k^i has three predecessor tasks t_1^i, t_2^i and t_3^i as shown in Fig. 2. It is assumed that all tasks are mapped to the same VM instance type VM $vm_s^{vt_u}$ and each task's execution time is 20m, 30m, and 60m

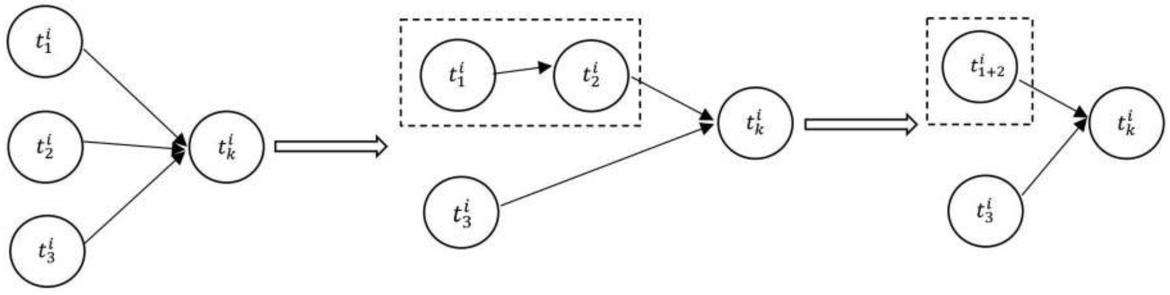


Fig. 2. The process of predecessor tasks merging.

respectively. Task t^i_k can not start its execution until the task t^i_3 finish its execution since the task t^i_3 has the longest execution time . However, if we change the relationship between t^i_1 and t^i_2 as a sequence as shown in Fig. 2, the total execution time of t^i_1 and t^i_2 is less than 50m and not more than 60m. Therefore, if t^i_1 and t^i_2 tasks are merged, it helps to reduce the number of VMs used, prevents the data transfer time between t^i_1 and t^i_2 when they run on different resources and also decreases the run time overhead.

5.2. Workflow pre-processing

When a user submits a new workflow w_i to the scheduler, Workflow Pre-Processor merges the pipeline tasks and determines the earliest start time EST^i_j and latest finish time LFT^i_j for each task in a workflow w_i. Then all the tasks are categorized as ready and waiting tasks. Tasks without any predecessors are called as ready tasks and are added to the ready task pool (*readyTaskPool*), otherwise, the tasks are called as waiting tasks and are added to the task pool (*taskPool*). The workflow Pre-Processor algorithm is presented in Algorithm 2.

Algorithm 2: Workflow Pre-Processor

```
Input  : Set of Workflows
Output: readyTaskPool, taskPool
$taskPool \leftarrow NULL$
$readyTaskPool \leftarrow NULL$
foreach workflow $w_i \in W$ do
    Call the task merging algorithm
    foreach task $t^i_j \in w_i$ do
        calculate $EST^i_j$ and $LFT^i_j$ using equation (1) and (4)
        if $pred\left(t^i_j\right)$ is NULL then
            $readyTaskPool \leftarrow readyTaskPool \cup \left\{t^i_j\right\}$
        else
            $taskPool \leftarrow taskPool \cup \left\{t^i_j\right\}$
        end
    end
end
```

5.3. WQ task processor

WQ Task Processor moves the waiting tasks to the ready task pool. The strategy for moving the tasks from *taskPool* to *readyTaskPool* is presented in Algorithm 3. For each task in the *taskPool* , whose EST^i_j equals the current time $Time_{current}$ and predecessors of the task has completed their execution are removed from the *taskPool* and added to the *readyTaskPool*. After that, all tasks in *readyTaskPool* are sorted in ascending order of the latest start time.

Algorithm 3: WQ Task Processor

```
Input  : taskPool
Output: Updated readyTaskPool and taskPool
while taskPool is not empty do
    foreach task $t^i_j \in taskPool$ do
        if $EST^i_j = Time_{current}$ and Predecessors of $t^i_j$ has completed thier execution then
            $readyTaskPool \leftarrow readyTaskPool \cup \left\{t^i_j\right\}$
            $taskPool \leftarrow taskPool - \left\{t^i_j\right\}$
        end
    end
    Sort readyTaskPool by earliest start time in ascending order
    call function TaskScheduler();
end
```

5.4. Task Scheduler

Task Scheduler attempts to execute the task t^i_j at minimal cost before its latest finish time LFT^i_j. First, for all the set of idle VMs, TOPSIS algorithm is called to get the optimal VM which can finish the task t^i_j before LFT^i_j with minimal cost and task will be assigned. If idle VM is not available then the TOPSIS algorithm is called for all the VM types in cloud to find a VM type such that it meets the task's latest finish time with minimum cost. A new VM will be acquired to execute the task.

Algorithm 4: TaskScheduler

```
Input  : Task
Output: Optimal VM for task
$VM_{selected} \leftarrow NULL$
$VM_{selected} \leftarrow CallTOPSIS(Task, AllidleVMs)$
if $VM_{selected} == NULL$ then
    $VM_{selected} \leftarrow CallTOPSIS(Task, AllVMTypes)$
    Acquire a new VM of type $VM_{type}$ , and schedule task $t^i_j$
else
    Schedule task $t^i_j$ to the selected VM $VM_{Sel}$
end
```

5.5. Task monitor

Task Monitor constantly collects the task state information and if any tasks complete its execution then it recalculates the EST^i_j and LFT^i_j for the successor tasks. For each successor, it gets all the predecessors' task information and If all the predecessors are executed then the task is moved to the ready task pool from the task pool. The task monitor algorithm is presented in Algorithm 5.

Algorithm 5: TaskMonitor

```
Input  : ExecutedTask
Output: Updated readyTaskPool and taskPool
foreach successor task $t^i_s \in$ succ $\left(t^i_j\right)$ do
    calculate $EST^i_j$ and $LFT^i_j$ using equation (1) and (4)
    if pred $\left(t^i_s\right)$ is executed then
        $readyTaskPool \leftarrow readyTaskPool \cup \left\{t^i_j\right\}$
        $taskPool \leftarrow taskPool - \left\{t^i_j\right\}$
    end
end
```

6. Performance evaluation

This section presents a comparison of the T-CCWSA, with recently published Cloud-based Workflow Scheduling Algorithm (CWSA) (Rimal and Maier, 2017), Earliest Finish Time – Maximum Effective Reduction (EFT-MER) (Lee et al., 2015) and Heterogeneous Earliest-Finish-Time (HEFT) (Topcuoglu et al., 2002) that match our goal and conditions. These algorithms are briefly explained below.

The cloud-based Workflow Scheduling Algorithm (CWSA) is designed for online scheduling. When a new workflow arrives, computing resources are sorted by their computational speeds in descending order. To minimize the cost of execution, this algorithm inserts workflow tasks into idle slots of the resources. If it is not feasible, the minimum completion time approach is applied to schedule the tasks.

Earliest Finish Time – Maximum Effective Reduction (EFT-MER) algorithm applies the earliest finish time (EFT) to generate the base execution plan for all the workflow tasks. Further, the Maximum Effective Reduction (MER) refines the base execution plan by merging the resources with less workload to fill idle time slots on other resources.

HEFT algorithm assigns an upward rank to each workflow task based on the maximal length to the exit task. Subsequently, it selects the task with the highest upward rank and assigns it to the resources with minimum finish time.

To evaluate the performance impact of algorithms, we consider 1. Total cost as in Eq. (9) 2. Resource Utilization as in Eqs. (10) and 3. Planning Success Rate (PSR) expressed by Eq. (12) and defined in Zheng and Sakellariou (2013) and Zheng and Sakellariou (2012). The PSR provides the percentage of valid schedules obtained in a given experiment.

$$PSR = 100 \times \frac{\text{Number of simulation runs that successfully meet deadline}}{\text{Total number of simulation runs}} \quad (12)$$

6.1. Experimental workflows

The performance of T-CCWSA is evaluated with different workflows used in different scientific fields: LIGO, Epigenomics, CyberShake, and Montage. These workflows have diverse structural properties such as pipeline, aggregation, distribution, and redistributions as well as different composition as shown in Fig. 3. Montage, an astronomy application stitches a series of images to create personalized sky mosaics. Montage tasks require high intense I/O and CPU with low processing capacity. The LIGO workflow aims to detect gravitational waves. This workflow requires a large memory with a high CPU. The Epigenomics is used in bioinformatics to automate genome sequencing operation. Moreover, these tasks demand high power computational processors with limited I/O regulations. Cybershake is best suited for simulating the earthquake hazards using synthetic seismograms. These workflow tasks require large memory and high CPU. To ease the evaluation of scheduling algorithms, Bharathi et al. (2008) developed a set of synthetic workflows of various sizes that resembles concrete scientific workflows. Synthetic workflows are characterized by DAG in XML format and are available in Workflow Generator. For assessing the results of the proposed algorithm in terms of performance, experiments are carefully designed and carried out for the above-specified workflows with the varying number of tasks: small (about 25 tasks), average (about 100 tasks) and large (about 1000 tasks).

6.2. Experimental settings

The cloud service providers provide various types of VMs with varying configurations. The VM configurations of EC2 cloud offerings (Anwar and Deng, 2018) are shown in Table 2. It is assumed that for each type of VM, the processing capacity in terms of floating-point operations per second (FLOPS) is available from the provider or can be estimated (Schad et al., 2010). The estimated time of execution of workflow tasks in various types of virtual machines is obtained based on their processing capacity. The change in CPU performance of each VM is modeled based on the results presented by Schad et al. (2010). Also, each virtual machine's performance is reduced by a maximum of 24% based on the normal distribution. Its average mean is found to be 12% along with a 10% standard deviation. Similarly, the data transfer time in the same data center is increased by a maximum of 19% (Schad et al., 2010), based on the normal distribution. Its average mean is found to be 9.5% along with a 5% standard deviation. The average bandwidth is set based on Amazon's Elastic Block Store (Amazon Elastic Block Store) i.e., 20 MBps. The VM billing time is set to 10-minute interval and the estimated acquisition delay is set to one minute similar to Meena et al. (2015).

To evaluate the performance metrics, the deadline factor *BaseDeadline* is considered. We introduce a deadline factor δ similar to Abrishami et al. (2013), based on which we vary the deadlines for workflows. We vary δ from 0.3 to 1.5 with a step length of 0.3. For this, all workflow tasks are executed on the fastest resources and the minimum time to run the workflow W^i_{MET} is obtained. W^i_{MET} is the lower limit of the i^{th} workflow execution time. The deadlines are established according to the rule specified in Eq. (13).

$$D_{w_i} = T^i_a + W^i_{MET} * (1 + \delta) \quad (13)$$

where T^i_a is the workflow arrival time, W^i_{MET} is the minimum time required to execute the submitted workflow and δ is the deadline factor defined as follows. For '*BaseDeadline*': $0.3 \leqslant \delta \leqslant 1.5$

The deadline factor δ was varied with 0.3 disparity

Table 2
VM Instance specifications.

Type of VM	ECU	Memory(GiB)	Cost($/h)
m3.medium	1	3.75	0.067
c3.xlarge	4	3.75	0.21
m3.xlarge	4	15	0.266
c3.2xlarge	8	15	0.42
m3.2xlarge	16	30	0.532

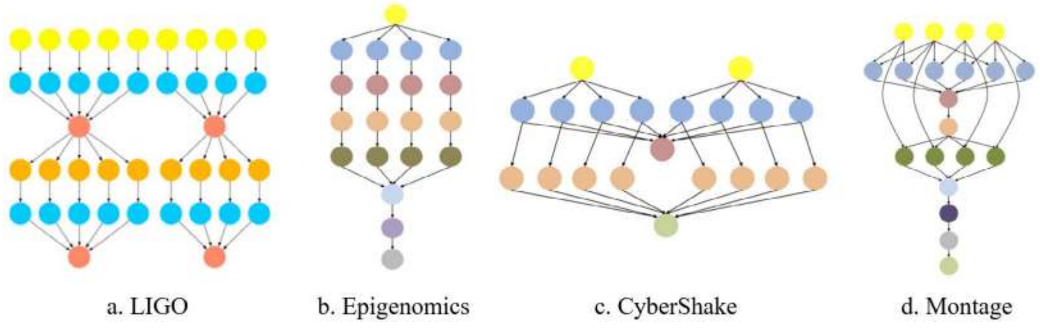


Fig. 3. Workflow structures differed in terms of characterization.

6.3. Performance impact of workflow deadline

Fig. 4 presents the performance impact of workflow deadline of the proposed T-CCWSA as well as existing algorithms – CWSA, EFT-MER, and HEFT. To evaluate the performance of the algorithms across varying workflow deadlines, BaseDeadline is gradually increased from 0 to 1.5 with an increment of 0.3.

It can be seen in Fig. 4a that when the deadline factor increases, the cost of the four algorithms descends slightly. It is because extending workflow deadline enables more laxity time to complete workflows execution within their deadline. Therefore, more parallel tasks can be assigned to the same virtual machine for execution, so that fewer VMs are used. Further, VMs with low configurations and pricing can also meet workflows' deadlines, which further reduces the overall cost. The experimental results in Fig. 4a shows that the cost of T-CCWSA is less than CWSA, EFT-MER, and HEFT on average by 9.23%, 11.92%, and 26.54% respectively. The following facts can be attributed to the superiority of the proposed T-CCWSA. First, the T-CCWSA algorithm schedules tasks to virtual machines only when they are ready. Next merging of pipeline tasks reduces the execution cost of tasks as we do not need to transfer the dataset.

Fig. 4b shows that the utilization of T-CCWSA, CWSA, EFT-MER and HEFT algorithms has a rising trend with an increase in BaseDeadline. This is because when the BaseDeadline is increased, the deadlines of the workflows are longer, which helps to execute more parallel tasks consecutively on the same resource, thus reducing the idle time slots. Also, the average resource utilization of the proposed T-CCWSA is 4.8%, 11.8%, and 22% higher than CWSA, EFT-MER, and HEFT respectively. T-CCWSA's high resource utilization is reasonable because there is no data dependency among the ready tasks assigned to virtual machines for their execution; hence, idle time slots on resources can be reduced as much as possible. But in other comparative algorithms, all the workflow tasks are immediately mapped to the virtual machines, which limit such algorithms to schedule tasks based on run time information.

6.4. Performance impact of workflow count

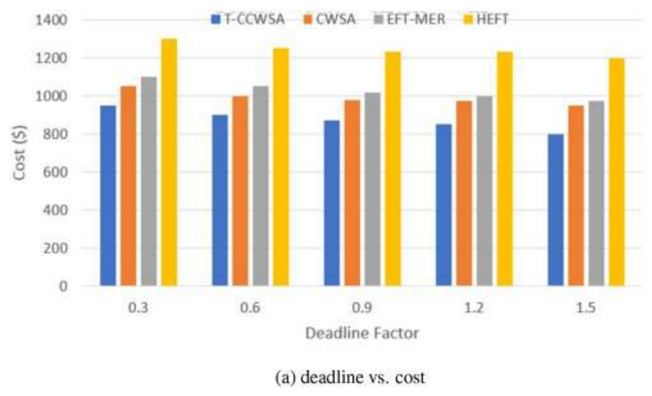
To evaluate the performance of the four algorithms in the context of workflow count, we gradually increase the workflow count from 500 to 2500 with an increment of 500, while fixing the deadline base. The experimental results of T-CCWSA, CWSA, EFT-MER, and HEFT are illustrated in Fig. 5.

It can be seen from Fig. 5a, the total execution cost increases linearly with the count of workflows. The reason is evident because more workflows involve more VMs with longer working time, resulting in higher costs. Furthermore, the experimental results exhibit that the T-CCWSA's total cost is less than CWSA, EFT-MER, and HEFT on average by 5.26%, 6.57%, and 20.52%, respectively.

From Fig. 5b, we can observe that the resource utilization of algorithm T-CCWSA, CWSA, EFT-MER and HEFT is stable at 79.40%, 74.65%, 67.60%, and 57.40%, regardless of the change of the workflow count. These experimental results demonstrate that the proposed T-CCWSA algorithm has significant advantages in improving resource utilization of cloud resources.

6.5. Planning success rate

Fig. 6a shows the results obtained for CYBERSHAKE at different combinations of deadline factor. When the deadline factor is



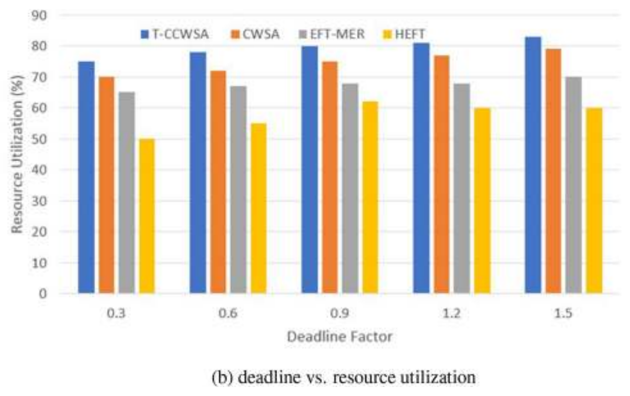
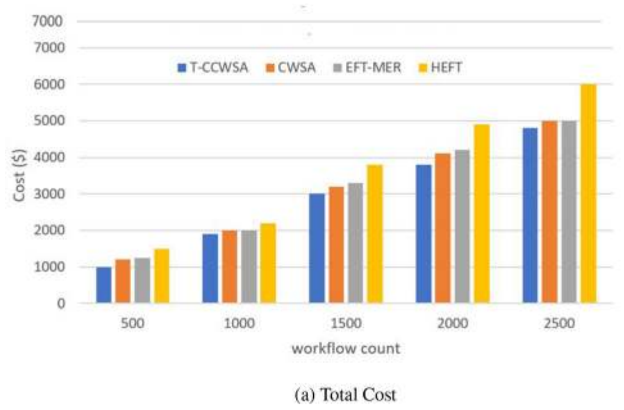


Fig. 4. Performance impact of workflow deadline.



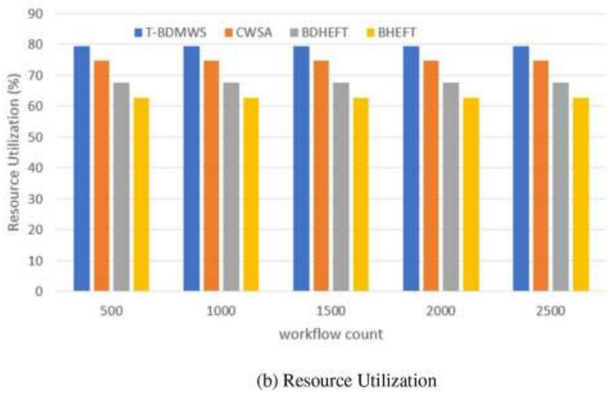
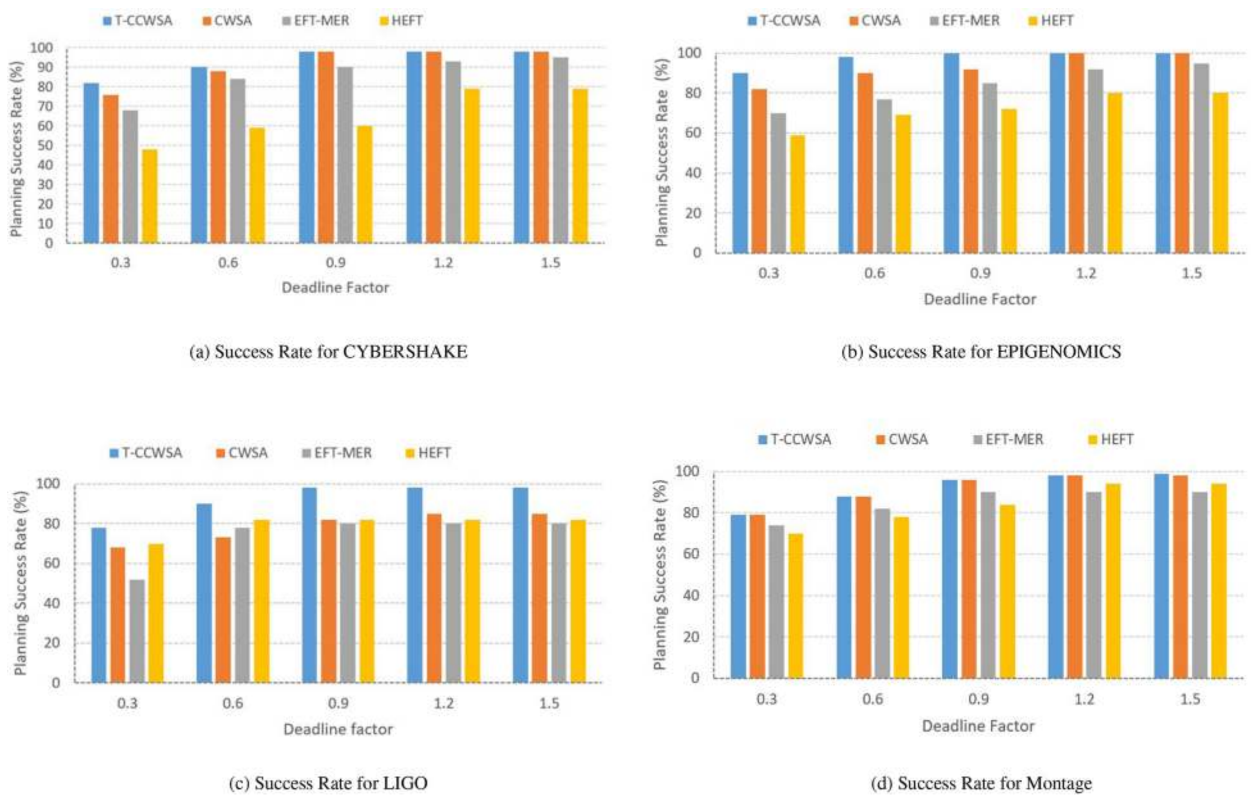


Fig. 5. Performance impact of workflow count.



(a) Success Rate for CYBERSHAKE (b) Success Rate for EPIGENOMICS (c) Success Rate for LIGO (d) Success Rate for Montage

Fig. 6. Planning Success Rate.

relatively small, the average Planning Success Rate (PSR) of the T-CCWSA algorithm is relatively high. When the deadline factor is increased, the PSR of the T-CCWSA algorithm is consistent with the CWSA algorithm. Compared with that in other algorithms, the PSR of the HEFT algorithm is less.

Fig. 6(b–d) show the PSR obtained for the EPIGENOMICS, LIGO and Montage applications. As seen from Fig. 6b–d among CWSA, EFT-MER, HEFT, and T-CCWSA, the proposed algorithm obtains good performance for the range of deadline values. By increasing the deadline factor, more laxity time is available to execute the workflow resulting in an increase in the PSR value for T-CCWSA. The reason for the better performance of the T-CCWSA algorithm is, it schedules tasks to virtual machines only when they are ready. As a result, it lowers the schedule length of the workflow tasks. Therefore, it creates a schedule with a lower cost within the defined deadline.

6.6. Discussion

The above results show that the proposed algorithm has a superior performance over the baseline algorithms CWSA, EFT-MER, and HEFT. The intuition behind such performance superiority is as follows:

- The T-CCWSA algorithm schedule tasks to virtual machines only when they are ready.
- Data transfer cost is reduced by merging the pipeline tasks.
- T-CCWSA's high resource utilization is reasonable because only the ready tasks are dynamically scheduled to VM and the idle time slots are cut down as much as possible.
- The performance of the T-CCWSA is enhanced by implementing MCDM, TOPSIS method as a decision maker. The TOPSIS method finds the ideal resource among the available resources by considering three criteria as execution time, execution cost and data transfer time of the tasks.

7. Conclusion

Most of the existing concurrent workflow scheduling algorithms have focused on offline scheduling; Only very few addressed online scheduling. This paper presents an algorithm called T-CCWSA (TOPSIS inspired Cost-Efficient Concurrent Workflow Scheduling Algorithm) to reduce the cost and improve the resource utilization for cloud platforms under user-defined deadline constraint for dynamic concurrent workflows. The Proposed T-CCWSA makes a good make good trade-off between cost, system's resource utilization. For simulation, the CloudSim tool is used and the obtained results are compared with baseline algorithms such as CWSA, EFT-MER, and HEFT on diverse real-world scientific workflows. Observations reveal that the presented scheme provides better results in terms of cost-effective realistic schedules. As future work, this algorithm can be extended by considering other QoS parameters such as energy, security, reliability and multiple pricing schemes from various cloud service providers. Therefore, the combination of these issues opens a direction of our future works. In addition, workflow scheduling is part of classical multi-objective optimization problem, and applying multi-objective evolutionary algorithms to solve it is another concern.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Abdel-Basset, M., Mohamed, M., Chang, V., 2018. NMCDA: a framework for evaluating cloud computing services. Future Gener. Comput. Syst. 86, 12–29. https://doi.org/10.1016/j.future.2018.03.014.

Abdel-Basset, M., Manogaran, G., Mohamed, M., Chilamkurti, N., 2018. Three-way decisions based on neutrosophic sets and AHP-QFD framework for supplier selection problem. Future Gener. Comput. Syst. 89, 19–30. https://doi.org/10.1016/j.future.2018.06.024.

Abdel-Basset, M., Gunasekaran, M., Mohamed, M., Chilamkurti, N., 2019. A framework for risk assessment, management and evaluation: economic tool for quantifying risks in supply chain. Future Gener. Comput. Syst. 90, 489–502. https://doi.org/10.1016/j.future.2018.08.035.

Abrishami, S., Naghibzadeh, M., Epema, D.H., 2012. Cost-driven scheduling of grid workflows using partial critical paths. IEEE Trans. Parallel Distrib. Syst. 23 (8), 1400–1414. https://doi.org/10.1109/tpds.2011.303.

Abrishami, S., Naghibzadeh, M., Epema, D.H., 2013. Deadline-constrained workflow scheduling algorithms for Infrastructure as a Service Clouds. Future Gener. Comput. Syst. 29 (1), 158–169. https://doi.org/10.1016/j.future.2012.05.004.

Amazon Elastic Block Store (EBS) – Amazon Web Services. (n.d.). Retrieved fromhttp://aws.amazon.com/ebs/..

Anwar, N., Deng, H., 2018. Elastic scheduling of scientific workflows under deadline constraints in cloud computing environments. Future Internet 10 (1), 5. https://doi.org/10.3390/fi10010005.

Arabnejad, H., Barbosa, J., 2012. Fairness Resource Sharing for Dynamic Workflow Scheduling on Heterogeneous Systems. 2012 IEEE 10th International Symposium on Parallel and Distributed Processing with Applications. https://doi.org/10.1109/ispa.2012.94..

Arabnejad, H., Barbosa, J.G., 2014. Budget Constrained Scheduling Strategies for On-line Workflow Applications. Computational Science and Its Applications – ICCSA 2014 Lecture Notes in Computer Science, pp. 532–545.https://doi.org/10.1007/978-3-319-09153-2_40..

Arabnejad, H., Barbosa, J.G., 2014. List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Trans. Parallel Distrib. Syst. 25 (3), 682–694. https://doi.org/10.1109/tpds.2013.57.

Arabnejad, H., Barbosa, J.G., 2014. A budget constrained scheduling algorithm for workflow applications. J. Grid Comput. 12 (4), 665–679. https://doi.org/10.1007/s10723-014-9294-7.

Arabnejad, H., Barbosa, J.G., 2015. Multi-workflow QoS-Constrained Scheduling for Utility Computing. 2015 IEEE 18th International Conference on Computational Science and Engineering.https://doi.org/10.1109/cse.2015.29..

Arabnejad, H., Barbosa, J.G., 2017. Maximizing the completion rate of concurrent scientific applications under time and budget constraints. J. Comput. Sci. 23, 120–129. https://doi.org/10.1016/j.jocs.2016.10.013.

Arabnejad, H., Barbosa, J.G., Suter, F., 2014. Fair resource sharing for dynamic scheduling of workflows on heterogeneous systems. High-Performance Computing on Complex Environments 145–167. https://doi.org/10.1002/9781118711897.ch9.

Arabnejad, V., Bubendorfer, K., Ng, B., 2019. Budget and deadline aware e-science workflow scheduling in clouds. IEEE Trans. Parallel Distrib. Syst. 30 (1), 29–44. https://doi.org/10.1109/tpds.2018.2849396.

Assunção, M.D.D., Costanzo, A.D., Buyya, R., 2010. A cost-benefit analysis of using cloud computing to extend the capacity of clusters. Cluster Comput. 13 (3), 335–347. https://doi.org/10.1007/s10586-010-0131-x.

Azad, P., Navimipour, N.J., 2017. An energy-aware task scheduling in the cloud computing using a hybrid cultural and ant colony optimization algorithm. Int. J. Cloud Appl. Comput. 7 (4), 20–40. https://doi.org/10.4018/ijcac.2017100102.

Aziz, M.A., Abawajy, J., Herawan, T., 2015. Layered workflow scheduling algorithm. In: 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). https://doi.org/10.1109/fuzz-ieee.2015.7338091.

Behzadian, M., Otaghsara, S.K., Yazdani, M., Ignatius, J., 2012. A state-of the-art survey of TOPSIS applications. Expert Syst. Appl. 39 (17), 13051–13069. https://doi.org/10.1016/j.eswa.2012.05.056.

Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M., & Vahi, K., 2008. Characterization of scientific workflows. 2008 Third Workshop on Workflows in Support of Large-Scale Science.https://doi.org/10.1109/works.2008.4723958..

Bulgurcu, B., 2012. Application of TOPSIS technique for financial performance evaluation of technology firms in Istanbul stock exchange market. Proc. – Soc. Behav. Sci. 62, 1033–1040. https://doi.org/10.1016/j.sbspro.2012.09.176.

Calheiros, R.N., Buyya, R., 2014. Meeting deadlines of scientific workflows in public clouds with tasks replication. IEEE Trans. Parallel Distrib. Syst. 25 (7), 1787–1796. https://doi.org/10.1109/tpds.2013.238.

Capuano, N., Loia, V., Orciuoli, F., 2017. A fuzzy group decision making model for ordinal peer assessment. IEEE Trans. Learn. Technol. 10 (2), 247–259. https://doi.org/10.1109/tlt.2016.2565476.

Capuano, N., Chiclana, F., Fujita, H., Herrera-Viedma, E., Loia, V., 2018. Fuzzy group decision making with incomplete information guided by social influence. IEEE Trans. Fuzzy Syst. 26 (3), 1704–1718. https://doi.org/10.1109/tfuzz.2017.2744605.

Carrasco, R.A., Sánchez-Fernández, J., Muñoz-Leiva, F., Blasco, M.F., Herrera-Viedma, E., 2015. Evaluation of the hotels e-services quality under the user's experience. Soft. Comput. 21 (4), 995–1011. https://doi.org/10.1007/s00500-015-1832-0.

Chang, V., Abdel-Basset, M., Ramachandran, M., 2018. Towards a reuse strategic decision pattern framework – from theories to practices. Inf. Syst. Front. 21 (1), 27–44. https://doi.org/10.1007/s10796-018-9853-8.

Chen, Y., Kilgour, D.M., Hipel, K.W., 2008. Screening in multiple criteria decision analysis. Decis. Support Syst. 45 (2), 278–290. https://doi.org/10.1016/j.dss.2007.12.017.

Cusumano, M., 2010. Cloud computing and SaaS as new computing platforms. Commun. ACM 53 (4), 27. https://doi.org/10.1145/1721654.1721667.

Daoud, M.I., Kharma, N., 2008. A high-performance algorithm for static task scheduling in heterogeneous distributed computing systems. J. Parall. Distrib. Comput. 68 (4), 399–409. https://doi.org/10.1016/j.jpdc.2007.05.015.

Deng, X., 2018. Analyzing the monotonicity of belief interval-based uncertainty measures in belief function theory. Int. J. Intell. Syst. 33 (9), 1869–1879. https://doi.org/10.1002/int.21999.

Emmanuel, B., Qin, Y., Wang, J., Zhang, D., Zheng, W., 2018. Cost optimization heuristics for deadline constrained workflow scheduling on clouds and their comparative evaluation. Concurrency and Computation: Practice and Experience 30 (20). https://doi.org/10.1002/cpe.4762.

Ghafouri, R., Movaghar, A., Mohsenzadeh, M., 2018. Time-cost efficient scheduling algorithms for executing workflow in infrastructure as a service clouds. Wireless Pers. Commun. 103 (3), 2035–2070. https://doi.org/10.1007/s11277-018-5895-y.

Guo, W., Lin, B., Chen, G., Chen, Y., Liang, F., 2018. Cost-driven scheduling for deadline-based workflow across multiple clouds. IEEE Trans. Netw. Serv. Manage. 15 (4), 1571–1585. https://doi.org/10.1109/tnsm.2018.2872066.

Hajji, M.A., Mezni, H., 2017. A composite particle swarm optimization approach for the composite SaaS placement in cloud environment. Soft. Comput. 22 (12), 4025–4045. https://doi.org/10.1007/s00500-017-2613-8.

He, Z., Jiang, W., 2018. An evidential dynamical model to predict the interference effect of categorization on decision making results. Knowl.-Based Syst. 150, 139–149. https://doi.org/10.1016/j.knosys.2018.03.014.

Hsu, C.-C., Huang, K.-C., Wang, F.-J., 2011. Online scheduling of workflow applications in grid environments. Future Gener. Comput. Syst. 27 (6), 860–870. https://doi.org/10.1016/j.future.2010.10.015.

Hwang, C.-L., Yoon, K., 1981. Methods for Multiple Attribute Decision Making: Methods and Applications Lecture Notes in Economics and Mathematical Systems, 58–191.https://doi.org/10.1007/978-3-642-48318-9_3..

Iyenghar, P., Pulvermueller, E., 2018. A model-driven workflow for energy-aware scheduling analysis of IoT-enabled use cases. IEEE IoT J. 5 (6), 4914–4925. https://doi.org/10.1109/jiot.2018.2879746.

Jatoth, C., Gangadharan, G., Buyya, R., 2017. Computational intelligence based QoS-aware web service composition: a systematic literature review. IEEE Trans. Serv. Comput. 10 (3), 475–492. https://doi.org/10.1109/tsc.2015.2473840.

Jiang, W., Hu, W., 2018. An improved soft likelihood function for Dempster-Shafer belief structures. Int. J. Intell. Syst. 33 (6), 1264–1282. https://doi.org/10.1002/int.21980.

Kang, B., Deng, Y., Hewage, K., Sadiq, R., 2019. A Method of measuring uncertainty for Z-number. IEEE Trans. Fuzzy Syst. 27 (4), 731–738. https://doi.org/10.1109/tfuzz.2018.2868496.

Kao, C., 2010. Weight determination for consistently ranking alternatives in multiple criteria decision analysis. Appl. Math. Model. 34 (7), 1779–1787. https://doi.org/10.1016/j.apm.2009.09.022.

Khorsand, R., Safi-Esfahani, F., Nematbakhsh, N., Mohsenzade, M., 2016. ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments. J. Supercomput. 73 (6), 2430–2455. https://doi.org/10.1007/s11227-016-1928-z.

Lee, Y.C., Han, H., Zomaya, A.Y., Yousif, M., 2015. Resource-efficient workflow scheduling in clouds. Knowl.-Based Syst. 80, 153–162. https://doi.org/10.1016/j.knosys.2015.02.012.

Lotfi, F.H., Rostamy-Malkhalifeh, M., Aghayi, N., Beigi, Z.G., Gholami, K., 2013. An improved method for ranking alternatives in multiple criteria decision analysis. Appl. Math. Model. 37 (1–2), 25–33. https://doi.org/10.1016/j.apm.2011.09.074.

Maheswaran, M., Ali, S., Siegal, H., Hensgen, D., Freund, R., 1999. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. Proceedings. Eighth Heterogeneous Computing Workshop (HCW99). https://doi.org/10.1109/hcw.1999.765094..

Martino, B.D., Cretella, G., Esposito, A., 2016. Cloud services composition through cloud patterns: a semantic-based approach. Soft. Comput. 21 (16), 4557–4570. https://doi.org/10.1007/s00500-016-2264-1.

Meena, J., Kumar, M., Vardhan, M., 2015. 'Efficient utilization of commodity computers in academic institutes: a cloud computing approach [Abstract]. Int. J. Comput., Electr., Automation, Control Inf. Eng. 9 (2).

Mir, M.A., Ghazvinei, P.T., Sulaiman, N., Basri, N., Saheri, S., Mahmood, N., et al., 2016. Application of TOPSIS and VIKOR improved versions in a multi criteria decision analysis to develop an optimized municipal solid waste management model. J. Environ. Manage. 166, 109–115. https://doi.org/10.1016/j.jenvman.2015.09.028.

Olson, D., 2004. Comparison of weights in TOPSIS models. Math. Comput. Modell. 40 (7–8), 721–727. https://doi.org/10.1016/j.mcm.2004.10.003.

Pan, L., Deng, Y., 2018. A new belief entropy to measure uncertainty of basic probability assignments based on belief function and plausibility function. Entropy 20 (11), 842. https://doi.org/10.3390/e20110842.

Partheeban, P., Kavitha, V., 2018. Versatile provisioning and workflow scheduling in WaaS under cost and deadline constraints for cloud computing. Trans. Emerg. Telecommun. Technol. 30 (1). https://doi.org/10.1002/ett.3527.

Rimal, B.P., Maier, M., 2017. Workflow scheduling in multi-tenant cloud computing environments. IEEE Trans. Parallel Distrib. Syst. 28 (1), 290–304. https://doi.org/10.1109/tpds.2016.2556668.

Sarraf, A.Z., Mohaghar, A., Bazargani, H., 2013. Developing TOPSIS method using statistical normalization for selecting knowledge management strategies. J. Ind. Eng. Manage. 6 (4). https://doi.org/10.3926/jiem.573.

Schad, J., Dittrich, J., Quiane-Ruiz, J., 2010. Runtime measurements in the cloud. Proceedings of the VLDB Endowment 3 (1–2), 460–471. https://doi.org/10.14778/1920841.1920902.

Shidpour, H., Shahrokhi, M., Bernard, A., 2013. A multi-objective programming approach, integrated into the TOPSIS method, in order to optimize product design; in three-dimensional concurrent engineering. Comput. Ind. Eng. 64 (4), 875–885. https://doi.org/10.1016/j.cie.2012.12.016.

Su, S.-F., Hsueh, Y.-C., Tseng, C.-P., Chen, S.-S., Lin, Y.-S., 2015. Direct adaptive fuzzy sliding mode control for under-actuated uncertain systems. Int. J. Fuzzy Logic Intell. Syst. 15 (4), 240–250. https://doi.org/10.5391/ijfis.2015.15.4.240.

Sun, R., Deng, Y., 2019. A new method to identify incomplete frame of discernment in evidence theory. IEEE Access 7, 15547–15555. https://doi.org/10.1109/access.2019.2893884.

Tian, G.-Z., Xiao, C.-B., Xu, Z.-S., Xiao, X., 2012. Hybrid scheduling strategy for multiple DAGs workflow in heterogeneous system. J. Software 23 (10), 2720–2734. https://doi.org/10.3724/sp.j.1001.2012.04198.

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. https://doi.org/10.1109/71.993206.

Wang, Y.-M., Luo, Y., 2010. Integration of correlations with standard deviations for determining attribute weights in multiple attribute decision making. Math. Comput. Modell. 51 (1–2), 1–12. https://doi.org/10.1016/j.mcm.2009.07.016.

Wang, N., Sun, Z., Su, S.-F., Wang, Y., 2018. Fuzzy uncertainty observer-based path-following control of underactuated marine vehicles with unmodeled dynamics and disturbances. Int. J. Fuzzy Syst. 20 (8), 2593–2604. https://doi.org/10.1007/s40815-018-0522-3.

Workflow Generator – Pegasus – Pegasus Workflow Management System. (n.d.). Retrieved fromhttps://confluence.pegasus.isi.edu/..

Wu, F., Wu, Q., Tan, Y., Li, R., Wang, W., 2016. PCP-B2: partial critical path budget balanced scheduling algorithms for scientific workflow applications. Future Gener. Comput. Syst. 60, 22–34. https://doi.org/10.1016/j.future.2016.01.004.

Xu, X., Xiao, C., Tian, G., Sun, T., 2017. Expansion slot backfill scheduling for concurrent workflows with deadline on heterogeneous resources. Cluster Comput. 20 (1), 471–483. https://doi.org/10.1007/s10586-017-0751-5.

Yager, R.R., 2018. Categorization in multi-criteria decision making. Inf. Sci. 460–461, 416–423. https://doi.org/10.1016/j.ins.2017.08.011.

Yang, X., Gao, J., Ni, Y., 2018. Resolution principle in uncertain random environment. IEEE Trans. Fuzzy Syst. 26 (3), 1578–1588. https://doi.org/10.1109/tfuzz.2017.2735941.

Yu, Z., Shi, W., 2008. A planner-guided scheduling strategy for multiple workflow applications. In: 2008 International Conference on Parallel Processing - Workshops. https://doi.org/10.1109/icpp-w.2008.10.

Yue, Z., 2011. A method for group decision-making based on determining weights of decision makers using TOPSIS. Appl. Math. Model. 35 (4), 1926–1936. https://doi.org/10.1016/j.apm.2010.11.001.

Yu, J., Buyya, R., Ramamohanarao, K., 2008. Workflow Scheduling Algorithms for Grid Computing. Studies in Computational Intelligence Metaheuristics for Scheduling in Distributed Computing Environments, 173–214.https://doi.org/10.1007/978-3-540-69277-5_7..

Zavadskas, E.K., Podvezko, V., 2016. Integrated determination of objective criteria weights in MCDM. Int. J. Inf. Technol. Decision Making 15 (02), 267–283. https://doi.org/10.1142/s0219622016500036.

Zavadskas, E.K., Bausys, R., Kaklauskas, A., Ubarte, I., Kuzminske, A., Gudiene, N., 2017. Sustainable market valuation of buildings by the single-valued neutrosophic MAMVA method. Appl. Soft Comput. 57, 74–87. https://doi.org/10.1016/j.asoc.2017.03.040.

Zhao, H., Sakellariou, R., 2006. Scheduling multiple DAGs onto heterogeneous systems. Proceedings 20th IEEE International Parallel & Distributed Processing Symposium. https://doi.org/10.1109/ipdps.2006.1639387.

Zheng, W., Sakellariou, R., 2012. Budget-deadline constrained workflow planning for admission control in market-oriented environments. Economics of Grids, Clouds, Systems, and Services Lecture Notes in Computer Science 105–119. https://doi.org/10.1007/978-3-642-28675-9_8.

Zheng, W., Sakellariou, R., 2013. Budget-deadline constrained workflow planning for admission control. J. Grid Comput. 11 (4), 633–651. https://doi.org/10.1007/s10723-013-9257-4.

Zhou, N., Qi, D., Wang, X., Zheng, Z., Lin, W., 2016. A list scheduling algorithm for heterogeneous systems based on a critical node cost table and pessimistic cost table. Concurrency and Computation: Practice and Experience 29 (5). https://doi.org/10.1002/cpe.3944.

Zhou, D., Al-Durra, A., Zhang, K., Ravey, A., Gao, F., 2018. Online remaining useful lifetime prediction of proton exchange membrane fuel cells using a novel robust methodology. J. Power Sources 399, 314–328. https://doi.org/10.1016/j.jpowsour.2018.06.098.

Zhou, N., Li, F., Xu, K., Qi, D., 2018. Concurrent workflow budget- and deadline-constrained scheduling in heterogeneous distributed environments. Soft. Comput. 22 (23), 7705–7718. https://doi.org/10.1007/s00500-018-3229-3.

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. https://doi.org/10.1109/tpds.2015.2446459.