J Supercomput (2018) 74:6529–6553
https://doi.org/10.1007/s11227-017-2043-5

Cloud workflow scheduling with hybrid resource provisioning

Long Chen[1] · **Xiaoping Li**[1]

Published online: 13 April 2017

Abstract Resource provisioning strategies are crucial for workflow scheduling problems which are widespread in cloud computing. The main challenge lies in determining the amounts of reserved and on-demand resources to meet users' requirements. In this paper, we consider the cloud workflow scheduling problem with hybrid resource provisioning to minimize the total renting cost, which is NP-hard and has not been studied yet. An iterative population-based meta-heuristic is developed. According to the shift vectors obtained during the search procedure, timetables are computed quickly. The appropriate amounts of reserved and on-demand resources are determined by an incremental optimization method. The utilization of each resource is balanced in a swaying way, in terms of which the probabilistic matrix is updated for the next iteration. The proposed algorithm is compared with modified existing algorithms for similar problems. Experimental results demonstrate effectiveness and efficiency of the proposed algorithm.

Keywords Resource allocation · Resources renting · Cost minimizing · Cloud computing

1 Introduction

Cloud computing is a new resources provision model in which resources more flexible than in distributed computing and grid computing. Users can access resources

✉ Xiaoping Li
xpli@seu.edu.cn

Long Chen
longc@seu.edu.cn

[1] Key Laboratory of Computer Network and Information Integration (Ministry of Education), Southeast University, Nanjing 211189, China

anywhere and anytime. Providers manage large amounts of resources (server, network, storage, platform, software, etc.). Users rent resources from providers when they require such services. Users' requests are commonly represented by workflows, which could describe a wide range of complex applications in distributed systems [13]. Generally workflow applications are supervised and executed in distributed or cloud systems, such as Pegasus [12], Askalon [33], Google MapReduce [10] and Amazon EC2 [20].

In cloud computing, the IaaS (infrastructure as a service) center usually provides three resource renting manners to the users: reserved, on-demand and spot [3]. The reserved manner is a long-term strategy which enables users to make a low, one-time payment for each demanded resource. Once the resource is reserved, it is owned during the entire period, i.e., the resource is not returned to the IaaS center until the due date. Users receive a significant discount for the resource renting, and the average cost could be reduced greatly. However, the rented resources are not fully used during the renting period which leads to poor resource utilization. On the contrary, the on-demand manner enables users to pay for computing capacity by hours without long-term commitments. The pay-on-demand leads to higher resource utilization. On-demand is the mostly adopted manner for workflow scheduling problems in cloud computing [2,6,9]. The spot manner enables users to bid for free resource capacity, e.g., the Amazon EC2 spot manner. Users bid for unused capacity. Instances are charged using the spot price, which is set by the provider and fluctuates periodically in terms of the supply and demand for resource capacity. Users' requests are met if the bid price is higher than the spot price. Instances are kept by the current user until new users come and make the spot price higher than the current bid price.

In this paper, we consider the workflow scheduling problems in cloud computing. Tasks are assumed to be non-preemptive and have long processing times, e.g., academic workflows. A non-preemptive task cannot be terminated until the execution of the task is finished. Tasks are constrained by complex precedence, and tasks are executed in parallel or sequentially. Therefore, the resource requirements are variable. The combination of the three provisioning manners can meet the variable requirements. The reserved manner decreases the cost greatly by the discount while the idle time intervals will be resulted. The on-demand manner saves cost by providing short-term resources for peak demands, whereas no discount is received. The spot manner [29] has the risk of failure (or out-of-bid) which is not allowed in the considered workflows. Therefore, we consider rent resources for variable requirements with both reserved and on-demand manners in this paper. To the best of our knowledge, there are only few papers about resource renting with both reserved and on-demand manners for the problem under study. Since the Resource Availability Cost Problem (RACP) adopting only the reserved manner, which is a special case of the considered problem, is NP-hard [23], it is not hard to prove that the considered problem is NP-hard. To determine the appropriate amounts of reserved and on-demand resources to meet variable requirements for the total renting cost minimization, we establish the integer programming model for the considered problem. Based on the Estimation of Distribution Algorithm (EDA) [24], which is commonly used population-based meta-heuristic for combinatorial optimization, an Adaptive Probabilistic Algorithm (APA) is developed.

The rest of the paper is organized as follows. Related works are described in Sect. 2. Section 3 establishes the mathematical model for workflow scheduling with hybrid resource provisioning. In Sect. 4, the proposed APA is described. Computational results are shown in Sect. 5, followed by conclusions in Sect. 6.

2 Related works

Scheduling workflow applications have been studied for many years. Workflow scheduling can be described as mapping tasks to suitable resources to satisfy some performance criterion. In the traditional distributed computing environments (e.g., utility Grids), resources are usually clusters geographically dispersed and encapsulated as services. Several services are candidates for each task of the workflow. Services are used exclusively, i.e., they are not shared by different tasks. There are two common objectives: cost optimization under deadline constraints or execution time optimization under budget constraints [35]. Common methods for time optimization include dynamic programming [16], branch and bound [17], decomposition-based methods [19], list scheduling algorithm [30], critical path-based allocation [25], greedy randomized adaptive search [4] and ant colony optimization approach [11]. The methods for cost optimization include the Deadline-MDP algorithm [36], Deadline Early Tree (DET) algorithm [37], Partial Critical Paths (PCP) algorithm [1] and the Critical Path-based Iterative (CPI) heuristic [8].

However, only a few papers have focused on scheduling workflow applications in cloud computing, in which resources (virtual machines) are usually geographically concentrated. Byun et al. [5] proposed a Balanced Time Scheduling (BTS) algorithm for allocating homogeneous resources to a workflow within a user-specified finish time. The on-demand manner is usually adopted in existing literature with the short-term resources provisioning for workflow applications. In the follow-up work of Byun et al. [6], a Partitioned Balanced Time Scheduling (PBTS) algorithm was proposed for scheduling the on-demand homogeneous resources to the workflow application. PBTS considers time partitions in the algorithm and minimizes the amount of resources for each time partition. Abrishami et al. [1] proposed a QoS-based Partial Critical Paths (PCP) workflow scheduling algorithm on utility grids, and the PCP algorithm was modified for the on-demand cases [2] in cloud computing. The two proposed algorithms in [2], IC-PCP and IC-PCPD2, are different from PCP in three aspects: on-demand resource provisioning, homogeneous networks and the pay-as-you-go pricing model. Cai et al. [7] considered the workflow scheduling problem with heterogeneous resources and on-demand resource provisioning. The problem was divided into two subproblems: service mapping and task tabling on sharable resources. Two heuristics Critical Path-based Iterative heuristic with Shortest services (CPIS) and List-based Heuristic considering Cost minimization and Mach degree maximization (LHCM) were developed for the subproblems, respectively. The on-demand manner is usually adopted for workflow applications in the existing literature.

Few papers concerned long-term resources provisioning in cloud computing which provide resources in the reserved manner. Workflow scheduling with reserved resources is similar to the Resource Availability Cost Problem (RACP) [14], a tradi-

tional project scheduling problem with scare time and unlimited amount of resources. RACP is an extension to the Resource-Constrained Project Scheduling Problem. The schedule of RACP determines the start time of each precedence-constrained task in the project to minimize the maximum resource amount during the whole project period. Common methods for RACP are exact algorithms [14,23,27] and meta-heuristics (e.g., Scatter Search [34], genetic algorithm [28] and path relinking [26]). RACP only considers resources provisioning with the reserved manner.

There are several studies on both the reserved and on-demand resource provisioning manners. From the cloud provider's viewpoint, Mazzucco et al. [22] attempted to maximize the net revenue of the cloud provider. Two resource provisioning manners were offered to different users: premium users with reserved instances and basic users with on-demand instances. The number of different instance types was determined by a queuing model. Chaisiri et al. [9] proposed an optimal cloud resource provisioning algorithm to get a balanced manner between the reserved and the on-demand. A stochastic programming model was formulated. The demand distribution of resources at each decision stage was supposed to be known in advance. Polynomial heuristics were proposed by Khatua et al. [22] for the hybrid resource provisioning problem. Van den Bossche et al. [32] presented a purchase management algorithm for the procurement decision on reserved contracts in the contexts of providers. However, these studies focused on independent task scheduling actually. These studies focused on independent task scheduling or virtual machine placement. To the best of our knowledge, no existing work exists on hybrid resource provisioning for workflow scheduling in which tasks are constrained by precedence.

3 Problem description

In this paper, we consider the workflow scheduling problems in cloud computing with long processing time and non-preemptive tasks. Tasks are fulfilled in parallel on multiple virtual machines, i.e., each task can be executed concurrently on several virtual machines. Resources are also shared by different tasks, i.e., a resource can be used by another task if the current task finishes during its resource renting period. Different types of resources (e.g., general-purpose, compute-optimized, GPU Instances, or memory-optimized virtual machine instances) are required by the tasks. Resources are provisioned with both reserved and on-demand manners.

A workflow application is usually denoted by an acyclic task-on-node network $G = (V, E)$. There are n nodes in $V = \{v_1, \ldots, v_n\}$. Each node $v_j \in V$ represents a task in the workflow, in which v_1 and v_n are the dummy source and sink tasks. Each edge $(v_i, v_j) \in E$ represents the precedence between task v_i and v_j. Heterogeneous virtual machines $R = \{R_1, \ldots, R_m\}$ are provided by the IaaS center and rent by consumers. Tasks are regarded as malleable tasks which can be executed on several machines in parallel. We suppose that the task v_j requires r_{jk} units of virtual machine R_k, and the processing time of v_j with these resources are d_j. All the processing time d_j (in hours) are supposed to be integer since virtual machines are charged by hour in this paper. There are two resources renting manners for users. Let the unit cost of virtual machine R_k $(k = 1, \ldots, m)$ with the on-demand manner be c_k. Let the

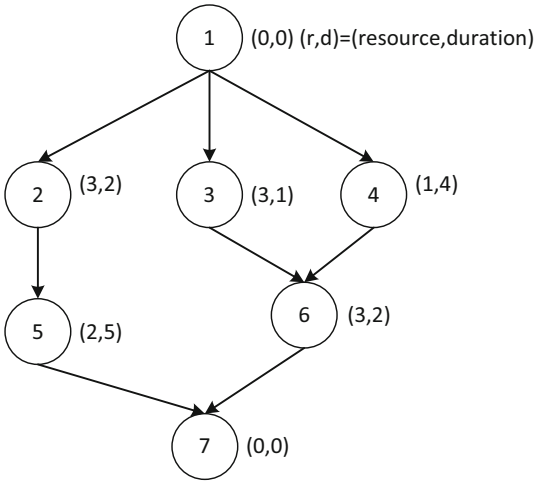


Fig. 1 An example of the considered problem

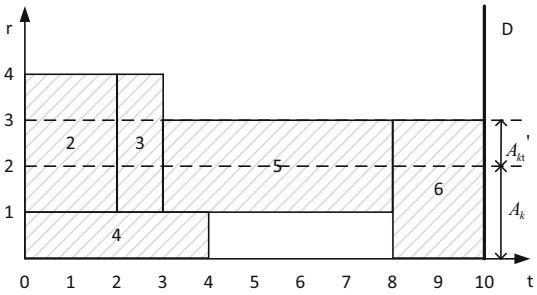


Fig. 2 An example and the schedule of the considered problem

discount of virtual machines with the reserved manner be φ. The unit cost of R_k with the reserved manner is $c_k \times \varphi$. Let the deadline of the workflow be D. The objective is to determine the minimum amount of reserved and on-demand virtual machines for the workflow with the deadline satisfied.

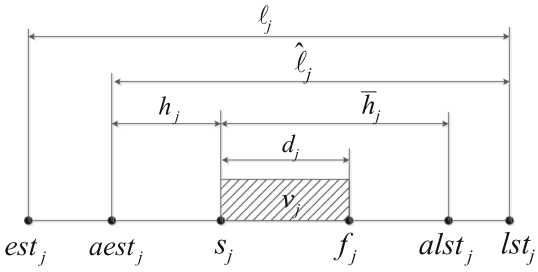
An example of the workflow application with 7 tasks and 1 type of resource is shown in Fig. 1. Nodes 1 and 7 are dummy tasks. The deadline of the workflow application is supposed to be 10 days. Let the unit cost of the on-demand resource be 5 and the discount of the reserved resource be 0.8. The required units of resource and the corresponding processing time are shown on the right of each task. Task 2 requires 3 units of resource with the processing time 2 days. Figure 2 shows a schedule for the workflow application, in which r is the number of virtual machines, D is the deadline of the workflow and t is the time (with the unit 1 day).

If all the 4 virtual machines are reserved, the total renting cost is $C = 4 \times 10 \times 0.8 \times 5 = 160$. If all the 4 virtual machines are on-demand, the total renting cost is $C = (4+4+4+3+2+2+2+2+3+3) \times 5 = 145$. If only virtual machines 2 and 3 are reserved, the other virtual machines are rented with the on-demand manner, the required on-demand resources are 2, 2, 2, 1, 0, 0, 0, 0, 1, 1 for each start time, respectively. The resource renting cost $C = 2 \times 10 \times 0.8 \times 5 + (2 + 2 + 2 +$

Table 1 Related parameters of v_j

Parameters	Notation
s_j	Start time
f_j	Finish time
d_j	Execution time
est_j	Earliest start time
lst_j	Latest start time
$aest_j$	Available earliest start time
$alst_j$	Available latest start time
h_j	Shift
$\bar{h}_j$	Compensate shift
ℓ_j	Slack time
$\hat{\ell}_j$	Available slack time

Fig. 3 Temporal parameters of v_j



$1 + 1 + 1) \times 5 = 125$. The hybrid resource provisioning manner saves the cost as compared to the only on-demand or reserved cases.

To determine the amount of resources for the workflow, the most important thing is to obtain the timetable $S = \{s_1, \ldots, s_n\}$ for the tasks, in which s_j is the actual start time of task v_j. Related parameters of v_j are shown in Table 1. Let the actual finish time of task v_j be f_j, i.e., $f_j = s_j + d_j$. Let est_j and lst_j be the earliest and latest start times of task v_j. $\mathcal{P}_j$ and $\mathcal{O}_j$ denote the immediate predecessor and immediate successor sets of v_j, i.e., $\mathcal{P}_j = \{v_i | (v_i, v_j) \in E)\}$, $\mathcal{O}_j = \{v_k | (v_j, v_k) \in E)\}$. By the critical path-based Forward and Backward Pass Calculations [15], parameters est_j, lst_j along with $\mathcal{P}_j$ and $\mathcal{O}_j$ of v_j ($v_j \in V$) can be obtained in $O(|E|)$. The available earliest start time (i.e., the earliest start time of an unscheduled task, given the start times of the scheduled activities) of task v_j is defined as $aest_j = \max_{v_i \in \mathcal{P}_j}\{f_i\}$. The available latest start time of task v_j is defined as $alst_j = \min_{v_k \in \mathcal{O}_j}\{s_k - d_j\}$. The shift h_j of task $v_j (j = 1, \ldots, n)$ represents the difference between s_j and $aest_j$, i.e., $h_j = s_j - aest_j$. Similarly, the compensate shift $\bar{h}_j$ is $\bar{h}_j = alst_j - s_j$. Let the slack time ℓ_j of v_j be $\ell_j = lst_j - est_j$. The available slack time $\hat{\ell}_j$ is defined as $\hat{\ell}_j = lst_j - aest_j$. Therefore, $0 \le h_j \le \hat{\ell}_j \le \ell_j$. Relationships of the temporal parameters of v_j are illustrated in Fig. 3.

Taking the task v_5 in Fig. 1 for example, the execution time d_5 is 5. Before allocating any tasks, the earliest and latest start time of v_5 are obtained with $est_5 = 3$ and $lst_5 = 5$. According to the schedule in Fig. 2, $s_5 = 3$ and $f_5 = 8$. The available

earliest and latest start times can be calculated by $aest_5 = \max_{v_i \in \mathcal{P}_5}\{f_i\} = f_2 = 2$ and $alst_5 = \min_{v_k \in \mathcal{O}_5}\{s_k - d_5\} = s_7 - 5 = 5$. According to these parameters, we obtain $h_5 = s_5 - aest_5 = 3 - 2 = 1, \bar{h}_5 = alst_5 - s_5 = 5 - 3 = 2, \ell_5 = lst_5 - est_5 = 5 - 2 = 3$ and $\hat{\ell}_5 = lst_5 - aest_5 = 5 - 3 = 2$, respectively.

Let π be a schedule of the considered problem. A_k is the amounts of R_k allocated to the workflow application with the reserved manner. A'_{kt} is the amounts of R_k allocated to the workflow application with the on-demand manner at time t. The total resource renting cost is calculated by $C(\pi) = \sum_{k=1}^{m}(f_n \times A_k \times c_k \times \varphi + \sum_{t=0}^{f_n} A'_{kt} \times c_k)$, in which f_n is the finish time of the dummy sink task. x_{jt} is a binary variable, which is 1 only if task v_j ($j \in \{1, \ldots, n\}$) starts at time t ($t \in \{0, \ldots, T\}$). Using the integer programming, workflow scheduling with hybrid resource provisioning is modeled as follows:

$$\min \sum_{k=1}^{m} \left(f_n \times A_k \times c_k \times \varphi + \sum_{t=0}^{f_n} A'_{kt} \times c_k \right) \tag{1}$$

s.t.

$$x_{jt} \in \{0, 1\},\ \forall j \in \{1, \ldots, n\},\ \forall t \in \{0, \ldots, D\} \tag{2}$$

$$A_k \geqslant 0,\ \forall k \in \{1, \ldots, m\} \tag{3}$$

$$A'_{kt} \geqslant 0,\ \forall k \in \{1, \ldots, m\},\ \forall t \in \{0, \ldots, D\} \tag{4}$$

$$\sum_{t=0}^{D} x_{jt} = 1,\ \forall j \in \{1, \ldots, n\} \tag{5}$$

$$\sum_{t=0}^{D} x_{it}t + d_i \leqslant \sum_{t=0}^{D} x_{jt}t,\ \forall j \in \{1, \ldots, n\}$$
$$\forall i \in \{1, \ldots, n\}\ i \neq j,\ \ \forall (v_i, v_j) \in E \tag{6}$$

$$\sum_{t=0}^{D} x_{nt}t + d_n \leqslant D, \tag{7}$$

$$A_k + A'_{kt} \geqslant \sum_{j=1}^{n} r_{jk} \sum_{\tau=t}^{t+d_j-1} x_{j\tau},$$
$$\forall k \in \{1, \ldots, m\},\ \forall t \in \{0, \ldots, D\} \tag{8}$$

Equation (2) denotes the binary variables for each task. x_{jt} equals to 1 if and only if the task v_j starts at time t. Formulas (3) and (4) ensure that the reserved or on-demand units of virtual machine R_k are nonnegative. Formula (5) ensures that each task starts exactly at one time. Precedence constraints of the tasks are denoted by Formula (6) with deadline constraints given in Formula (7). The resource availability constraints are established in Formula (8), in which $\sum_{\tau=t}^{t+d_j-1} x_{j\tau}$ is the task being processed at time t and $A_k + A'_{kt}$ is the amount of R_k available at time t.

4 Resource allocation with balanced scheme

The Adaptive Probabilistic Algorithm (APA) is proposed for the considered problem in this paper. APA is based on the Estimation of Distribution Algorithm (EDA) [24]. Unlike other evolutionary computing algorithms (e.g., genetic algorithm) producing new generations with explicit recombination operators, APA generates new populations implicitly, which makes APA different from usual recombination operators in EC. According to the probabilistic model of promising solutions, a solution distribution over the search space is estimated. A new generation is produced by sampling the search space using the estimated distribution. And vice versa, the estimated distribution is updated by the new generation.

There are five components in APA: shift vector generation scheme (SVGS), Shift Vector-based Timetabling (SVT), Incremental Renting Plan Decision (IRPD), Swaying Improvement Heuristic (SIH) and Weighted Voting-based Updating Mechanism. The flow chart of the proposed APA is shown in Fig. 4. By initializing all probabilities identical for the possible shift of each task, the initial probability distribution matrix **P** is obtained. In terms of the shift probability distribution matrix **P**, the shift vectors for population Pop are generated by SVGS. SVT is adopted to generate timetable for the schedules quickly. The amounts of reserved and on-demand resources are determined by IRPD. Some elites are selected from Pop and appended to the elite set $\mathcal{E}$. The selected elites are improved by the swaying heuristic SIH. WVUM estimates the probability distribution of the elites among the search space using a voting mechanism. **P** is updated by a learning rate-based mechanism. The next population is generated according to the updated **P**. The procedure is repeated until the termination criterion is satisfied. Let π^* be the best result of workflow scheduling with hybrid resource provisioning. Details of the four components of APA are given in Algorithm 1.

Algorithm 1: Adaptive Probabilistic Algorithm (APA)

```
1  begin
2      Initialize P, π*;
3      repeat
4          Generate Pop by SVGS;
5          Generate timetable for each π_i ∈ Pop by SVT;
6          Evaluate C(π_i) for each π_i ∈ Pop by IRPD;
7          Sort Pop in the non-decreasing order of C(π_i);
8          Select the first |E| elites from Pop by SIH;
9          Call BFMI to improve the elites in E;
10         π' ← the first element of E;
11         if C(π') < C(π*) then
12             π* ← π';
13         Estimate the probability distribution of E and update the distribution matrix P by WVUM;
14     until (Termination criterion is satisfied);
15     return π*;
```

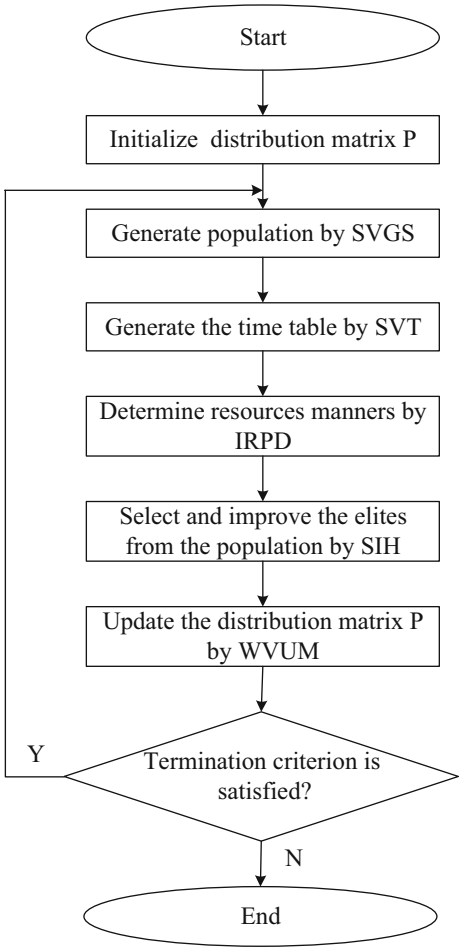


Fig. 4 Flow chart of APA

4.1 Shift vector generation scheme

The shift probability distribution matrix $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ is the foundation for generating shift vectors. According to the probability distribution matrix $\mathbf{P} = (\mathbf{p}_1, \ldots, \mathbf{p}_n)$ (see Fig. 5), shift vectors $\mathcal{H}$ are generated by the Shift Vector Generation Scheme (SVGS). For each task v_j, the shift h_j has at most ℓ_j states. Let p_{ji} $(j = 1, \ldots, n)$ be the probability that h_j takes the state i $(i = 0, \ldots, \ell_j)$. Each element of the probability vector $\mathbf{p}_j$ $(j = 1, \ldots, n)$ represents the probability of task v_j.

Theoretically, v_j could start at any time from the earliest start time est_j to the latest start time lst_j if there were no precedence between activities. However, v_j can start only after its predecessors have finished, i.e., s_j is located somewhere between $aest_j$ and lst_j because of the precedence constraints. In other words, to guarantee the feasibility of a schedule, h_j satisfies $h_j \in [0, \hat{\ell}_j]$ ($\hat{\ell}_j$ see Fig. 3) and p_{ji} is set as 0 for all $i = \hat{\ell}_j + 1, \ldots, \ell_j$. Initially, all probability vectors are uniformly distributed,

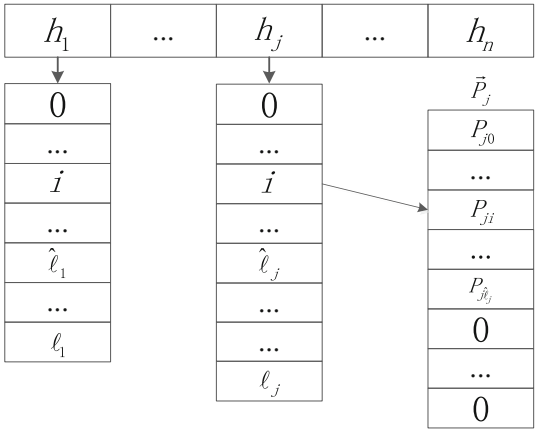


Fig. 5 Shift probability distribution matrix **P**

i.e., $\forall v_j \in V$, the probability is determined by $p_{ji} = \frac{1}{\ell_j+1}$. Starting from h_1, the shift vector $\mathcal{H}$ is iteratively generated by **P**. h_1 takes a random number uniformly from $[0, \ell_1]$ and $s_1 = h_1$. Obviously, $f_1 = s_1$ because v_1 is the dummy source node without any predecessor. Let $\mathcal{F}$ be the set of calculated tasks and $\mathcal{Q}$ be the list of ready ones, i.e., each element is an uncalculated task and all of its immediate predecessors have been processed. Initially, $\mathcal{F}$ is set as $\emptyset$ and $\mathcal{Q}$ is (v_1). The first element v_j is removed from $\mathcal{Q}$ and appended to $\mathcal{F}$. The new probability p'_{ji} of v_j is determined as follows.

$$p'_{ji} = \begin{cases} \frac{p_{ji}}{\sum_{i=0}^{\hat{\ell}_j} p_{ji}}, & \text{if } i = 0, \ldots, \hat{\ell}_j \\ 0, & \text{if } i = \hat{\ell}_j + 1, \ldots, \ell_j \end{cases}$$

Takes task v_5 in Figs. 1 and 2 for example, $\ell_5 = 3$ and $\hat{\ell}_5 = 2$. Initially, $\mathbf{p}_5 = (\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4})$, to make a feasible schedule, the new probability vector is set as $\mathbf{p}_5 = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0)$. Bigger deadline results in bigger slack time, and bigger shift vector of v_j. h_j takes a random number from $[0, \hat{\ell}_j]$ with probability p'_{ji}. For each immediate successor v_k of v_j, i.e., $(v_j, v_k) \in E$, v_k is appended to $\mathcal{Q}$ only when all of its immediate predecessors are included in $\mathcal{F}$. The process is repeated until there is no element in $\mathcal{Q}$. By this iterative procedure, the shift vector of the schedule is obtained. A feasible schedule is generated. SVGS is formally described in Algorithm 2.

4.2 Shift Vector-based Timetabling

Once a shift vector $\mathcal{H} = (h_1, \ldots, h_n)$ of a schedule is obtained from **P**, the start times of all tasks can be calculated by the following Shift Vector-based Timetabling (SVT) procedure. Let h_j be the shift of task v_j, $\mathcal{F}$ be the set of finished tasks. $\mathcal{X}$ is the list of a topological order of tasks $V = \{v_1, \ldots, v_n\}$, i.e., v_i is in front of v_j in $\mathcal{X}$ if v_i is a predecessor of v_j. For any schedule, $s_1 = 0$. In the schedule, the shift of a task is a delay that the task starts after all of its immediate predecessors finish. That is, the task does

Algorithm 2: Shift Vector Generation Scheme (SVGS)

```
1  begin
2      Pop ← ∅, Num ← 0;
3      repeat
4          F ← ∅, Q ← (v_1);
5          repeat
6              Take the first element v_j of Q;
7              for i = 0 to ℓ̂_j do
8                  p_ji ← p_ji / Σ_{i=0}^{ℓ̂_j} p_ji ;
9              for i = ℓ̂_j + 1 to ℓ_j do
10                 p_ji ← 0;
11             Generate a shift h_j from [0, ℓ_j] with probability p_ji;
12             s_j ← aest_j + h_j, f_j ← s_j + d_j;
13             F ← F ∪ {v_j}, Q ← Q − (v_j);
14             for each (v_j, v_k) ∈ E do
15                 if P_k ⊆ F then
16                     Q ← Q + (v_k);
17         until (Lenth(Q) = 0);
18         Pop ← Pop ∪ {(h_1, ..., h_n)}, Num ← Num + 1;
19     until (Num ≥ |Pop|);
20     return Pop;
```

not start immediately. Initially, $\mathcal{F}$ is set as $\emptyset$. The first element v_j is removed from $\mathcal{X}$ and appended to $\mathcal{F}$. s_j is calculated by $aest_j + h_j$. The process is repeated until there is no element in $\mathcal{X}$. The timetable of the schedule $S = \{s_1, \ldots, s_n\}$ is obtained. Take Fig. 1 for example, if the shift vector is $\mathcal{H} = (0, 0, 2, 0, 1, 4, 0)$ and the topological order is $\{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$, we can get the timetable $S = \{0, 0, 2, 0, 3, 8, 8\}$ directly with SVT without scheduling each task. SVT is formally described in Algorithm 3. In fact, different topological orders $\mathcal{X}$ produce the same timetable in SVT, which is guaranteed by the following theorem.

Algorithm 3: Shift Vector-based Timetabling (SVT)

```
1  begin
2      S ← ∅, s_1 ← 0, F ← {v_1}, X ← X − (v_1);
3      repeat
4          v_j ← the first element of X;
5          s_j ← aest_j + h_j;
6          F ← F ∪ {v_j}, X ← X − (v_j);
7          S ← S ∪ s_j;
8      until (Length(X) = 0);
9      return S;
```

Theorem 1 *SVT produces the same timetable for any topological order $\mathcal{X}$ of tasks $V = \{v_1, \ldots, v_n\}$.*

Proof For simplicity, we suppose two topological orders $\mathcal{X}^1 = (v_1, \ldots, v_i, \ldots, v_k, \ldots, v_n)$ and $\mathcal{X}^2 = (v_1, \ldots, v_k, \ldots, v_i, \ldots, v_n)$, they are different only on the positions of v_i and v_k. Let the timetable of $\mathcal{X}^1$ generated by SVT be $S^1 = \{s_1^1, \ldots, s_n^1\}$ and that of $\mathcal{X}^2$ be $S^2 = \{s_1^2, \ldots, s_n^2\}$.

For the tasks v_i and v_k (take v_k for example) $s_k^1 = aest_k^1 + h_k$, $s_k^2 = aest_k^2 + h_k$. Since h_k is given, it does not change during the calculation. So s_k is determined only by the start times of its predecessors. The fact that $\mathcal{X}^2$ is derived from $\mathcal{X}^1$ by just exchanging the positions of v_i and v_k, which means that there are no predecessors and successors of them between the position v_i and v_k. i.e., v_i (v_k) is not a predecessor or a successor of v_k (v_i). In other words, their start times are irrelevant to each other, i.e., $s_k^1 = s_k^2$, $s_i^1 = s_i^2$.

(1) For any task v_{j_1} before v_i in $\mathcal{X}^1$ (or before v_k in $\mathcal{X}^2$), v_k and v_i are not predecessors of v_{j_1} since $\mathcal{X}^1$ and $\mathcal{X}^2$ are topological orders. So $s_{j_1}^1 = s_{j_1}^2$.

(2) For any task v_{j_2} between v_i and v_k in $\mathcal{X}^1$ (or between v_k and v_i in $\mathcal{X}^2$), v_k and v_i are not predecessors of v_{j_2} according to the above analysis, i.e., $s_{j_2}^1 = s_{j_2}^2$.

(3) For any task v_{j_3} behind v_k in $\mathcal{X}^1$ (or behind v_i in $\mathcal{X}^2$), there are three cases. (i) Both v_k and v_i are not predecessors of v_{j_3}. It is obvious that $s_{j_3}^1 = s_{j_3}^2$. (ii) Either v_k or v_i is a predecessor of v_{j_3}, e.g., v_k is a predecessor of v_{j_3}. s_k is the same in both topological orders. If v_k is an immediate predecessor of v_{j_3}, its start time exerts no influence on those of the other immediate predecessors of v_{j_3}, hence $s_{j_3}^1 = s_{j_3}^2$. If v_k is not an immediate predecessor of v_{j_3}, v_k exerts no influence on the start times of nodes which are the immediate predecessors of v_{j_3} and are not successors of v_k. For immediate predecessors of v_{j_3} which are successors of v_k, we can obtain that the start time of v_{j_3} is not influenced by v_k with an iterative process, i.e., $s_{j_3}^1 = s_{j_3}^2$. (iii) Both v_k and v_i are predecessors of v_{j_3}. Since s_k and s_i are not changed, the start times of all the successors of v_k and v_i keep the same. Therefore, $s_{j_3}^1 = s_{j_3}^2$.

For task at any position of the topological order $\mathcal{X}^1$ and $\mathcal{X}^2$, $s_j^1 = s_j^2$. Thus, the timetable $S^1 = S^2$. Any other topological orders could be generated by exchanging positions of tasks in $\mathcal{X}^1$.

From the above discussion, therefore, it follows that SVT produces the same timetable for any topological order $\mathcal{X}$. □

Theorem 1 implies that for a given shift vector $\mathcal{H}$, no matter what topological order $\mathcal{X}$ has been used, the corresponding timetable S is unique. Obviously, the time complexity of SVT is $O(n)$. In comparison with the existing task sequence-based schedule representation methods ($O(n^2)$), this shift vector-based representation method is more intuitive without considering the complex precedence constraints.

Theorem 2 *A shift vector generated by SVGS is a feasible schedule.*

Proof Let $S = \{s_1, \ldots, s_n\}$ be the timetable generated by SVT in terms of $\mathcal{H} = (h_1, \ldots, h_n)$, a shift vector generated by SVGS. A schedule is impracticable if either the precedence constraints or the reserved deadline is not satisfied. However, (i) in Step 9 of SVGS, $s_j = aest_j + h_j$, $aest_j = \max_{v_i \in \mathcal{P}_j}\{f_i\}$ and $h_j \geq 0$, so $s_j \geq$

$s_i + d_i, \quad \forall (v_i, v_j) \in E$. The precedence constraints is satisfied. (ii) $h_n \leq \hat{\ell_n}$, so $s_n = aest_n + h_n \leq aest_n + \hat{\ell_n}$, and $s_n \leq lst_n \leq D$. The reserved deadline is satisfied. According to THEOREM 1, S is unique. In other words, for any task $v_j \in V = \{v_1, \ldots, v_n\}$, the start time s_j calculated by SVGS and that generated by SVT are the same. Therefore, a feasible schedule could be produced by a shift vector using SVGS. □

Theorem 2 illustrates that the feasible populations, Pop can be generated by the SVGS.

4.3 Incremental Renting Plan Decision

Resource renting plan is decided after SVT. The resource usage matrix $U = (u_{kt})_{m \times D}$ is calculated according to the timetable generated by SVT. Each element $u_{kt} \in U$ represents that u_{kt} units of virtual machine R_k are occupied at time t. For a timetable S, the corresponding matrix U is calculated as Algorithm 4. First, each element in U is set as 0, for each task v_j, if a virtual machine R_k is occupied, u_{kt} is updated by $u_{kt} + r_{jk}$ for the whole processing time of v_j.

Algorithm 4: Resource Usage Matrix

```
1 begin
2     for (k = 0;k < m;k ← k + 1) do
3         for (t = 0;t < D;t ← t + 1) do
4             u_kt ← 0;
5     for (j = 1;j < n + 1;j ← j + 1) do
6         for (k = 0;k < m;k ← k + 1) do
7             for (t = s_j;t < s_j + d_j;t ← t + 1) do
8                 u_kt ← u_kt + r_jk;
9     return U;
```

According to the resource usage matrix U, the amounts of reserved and on-demand resources are determined. Obviously, too many reserved resources lead to a low resource usage rate, which makes the total renting cost high, and too little reserved resources does not take the advantage of the discount in the long-term reserved resources, which also makes a high renting cost. To get a balance between the reserved and on-demand resources, the lower bounds are calculated first. The lower bounds for a virtual machine R_i are denoted as the minimal resource requirements in U, i.e., $Low_i = \min(u_{ij}), \forall j \in \{0, \ldots, D\}$. IRPD starts with $A_i = Low_i$, and u'_{ij} is set as $u_{ij} - A_i$. In the matrix $u'_{ij} (j \in \{0, \ldots, D\})$, the elements equal to 0 are called free time slots for virtual machine R_i. Let the total number of free time slots for virtual machine R_i be ζ. If the reserved amount A_i is increased by 1, some old free time slots ζ and on-demand resources $D - \zeta$ are calculated with the reserved way. The ζ free time slots increases the total renting cost by $\delta^+ = 1 \times \zeta \times c_i \times \varphi$ while the on-demand resources decrease the cost by $\delta^- = 1 \times c_i \times (D - \zeta) - 1 \times \varphi \times c_i \times (D - \zeta)$. If

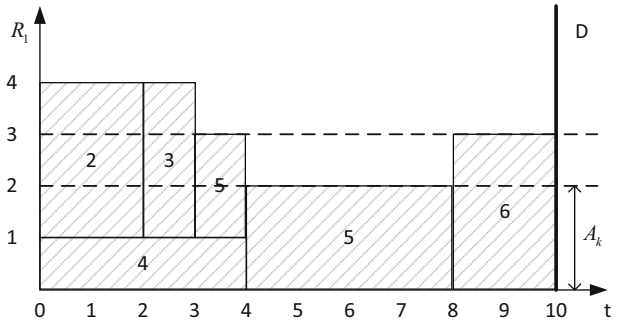


Fig. 6 An example of IRPD

$\delta^+ < \delta^-$, the increase in the reserved resource reduces the total renting cost, A_i is updated by $A_i + 1$. The procedure is repeated until $\varphi > (D - \zeta)/D$. The on-demand units for resource R_i is calculated by $A'_{ij} = u_{ij} - A_i$ for each time j. After getting the resource renting plan for each virtual machine, the total renting cost is calculated by Formula (1).

Figure 6 shows an example of IRPD. The matrix u_{1j} equals to (4, 4, 4, 3, 2, 2, 2, 2, 3, 3). A_i is set to be 2 first. $u'_{1j} = (2, 2, 2, 1, 0, 0, 0, 0, 1, 1)$. ζ and $D - \zeta$ are equal to 4 and 6, respectively. Suppose the discount φ be 0.8, i.e., $\varphi > (D - \zeta)/D$. Therefore, the procedure stops. $A_1 = 2$ and A'_{1j} is set to be (2,2,2,1,0,0,0,0,1,1), respectively.

4.4 Swaying Improvement Heuristic

All individuals in Pop are sorted in non-decreasing order of the resource renting cost. The first $|\mathcal{E}|$ schedules are selected. Usually, resources are always allocated in an unbalanced way in the initial schedule in terms of the probability-based methods. To balance the resource utilization, the swaying heuristic is developed. Different from traditional EDAs [24], the selected elites are improved by SIH before they are evaluated.

There are two procedures in SIH: Backward Moving and Forward Moving. Similar to h_j, there is a gap $\bar{h}_j$ between s_j and $alst_j$ of task v_j ($alst_j$ see Fig. 3). In the Backward Moving (BM), all tasks are sorted in the non-increasing order of the finish times of the current schedule and kept in the priority list $\mathcal{L}_B$. The head task $\mathcal{L}_B^{[1]}$ (the current task with the biggest finish time) denoted as $v_{[1]}$ is removed from $\mathcal{L}_B$ (the second one becomes the head task $\mathcal{L}_B^{[1]}$ now). There are $\bar{h}_{[1]}+1$ feasible start points for $v_{[1]}$ (from the current point $s_{[1]}$ to $alst_{[1]}$). Therefore, the new shift $h'_{[1]}$ is increased one by one from $h_{[1]}$ to $h_{[1]}+alst_{[1]}-s_{[1]}$. The corresponding resource renting costs of the possible schedules (only change the shift of $v_{[1]}$ while keeping the others unchanged) are calculated by IRPD. The shift vector (ties are broken by the larger shift) with the minimum cost is set as the new one. And the current $v_{[1]}$ is removed from $\mathcal{L}_B$. The procedure is repeated until $\mathcal{L}_B$ becomes empty. According to $\mathcal{L}_B$, all successors of $v_{[1]}$ have been calculated before the calculation of $v_{[1]}$. In other words, no immediate successor checking is needed. Forward Moving (FM) performs in an opposite way to BM. The priority list $\mathcal{L}_F$ is built according to the non-decreasing order of all the start

times of the schedule obtained by BM. And the decreasing strategy of shift vector is similar to the increasing one in BM, where the new shift $h'_{[1]}$ is decreased one by one from $s_{[1]} - aest_{[1]}$ to 0. The shift vector (ties are broken by the smaller shift) with the minimum cost is set as the new one. SIH starts from a schedule in $\mathcal{E}$ and iteratively conducts the BM and FM until no better resource renting cost can be found.

Let π^{best} be the best schedule found so far and π^c be the best solution of the current generation. For each schedule $\pi \in \mathcal{E}, BFMI(\pi)$ is described in Algorithm 5.

Algorithm 5: Backward-Forward Moving Improvement (SIH)

```
1  begin
2      π^c ← π;
3      repeat
4          L_B ← the non-increasing order of finish times of all tasks in π;
5          repeat
6              v_[1] ← L_B^[1];
7              Remove L_B^[1] from L_B;
8              π' ← π, h'_[1] ← h_[1];
9              repeat
10                 h'_[1] ← h'_[1] + 1;
11                 Calculate C(π') by IRPD;
12                 if C(π') ≤ C(π) then
13                     C(π) ← C(π'), h_[1] ← h'_[1];
14             until (h'_[1] > h_[1] + alst_[1] − s_[1]);
15         until (Lenth(L_B) = 0);
16         if C(π) < C(π^c) then
17             C(π^c) ← C(π), π^c ← π;
18         else
19             Break ;
20         L_F ← the non-decreasing order of start times of all tasks in π;
21         repeat
22             v_[1] ← L_F^[1];
23             Remove L_F^[1] from L_F;
24             π' ← π, h'_[1] ← s_[1] − aest_[1];
25             repeat
26                 h'_[1] ← h'_[1] − 1;
27                 Calculate C(π') by IRPD;
28                 if C(π') ≤ C(π) then
29                     C(π) ← C(π'), h_[1] ← h'_[1];
30             until (h'_[1] < 0);
31         until (Lenth(L_F) = 0);
32         if C(π) < C(π^c) then
33             C(π^c) ← C(π), π^c ← π;
34     until (C(π) > C(π^c));
35     return π^c;
```

4.5 Weighted Voting-based Updating Mechanism

The distribution estimation and the probability vector updating are crucial for APA to find good building blocks. The proposed WVUM estimates the probability distribution of the elites in the search space using a weighted voting mechanism. The probability vector **P** is updated by a learning rate-based mechanism.

Suppose π^{worst} is the worst solution in $\mathcal{E}$ and π^{best} is the best solution obtained so far. For each solution π with the resource renting cost $C(\pi)$ in $\mathcal{E}$, the weight $w(\pi)$ is denoted as

$$w(\pi) = \frac{C(\pi^{\text{worst}}) - C(\pi)}{C(\pi^{\text{best}})} \tag{9}$$

A binary variable is defined as

$$x_{jik} = \begin{cases} 1, & \text{if } i = h_j \text{ in schedule } \pi_k \\ 0, & \text{otherwise} \end{cases}$$

Since better schedule always implies better promising solution, the weight $w(\pi_k)$ is adopted to update the probability vector. Since the current probability exerts great influence on the performance of the algorithm, the probability p_{ji}^t in the tth generation is updated by

$$p_{ji}^{t+1} = (1-\beta)p_{ji}^t + \beta \frac{\sum_{\pi_k \in \mathcal{E}} x_{jik} \cdot w(\pi_k)}{\sum_{i=0}^{\ell_j} \sum_{\pi_k \in \mathcal{E}} x_{jik} \cdot w(\pi_k)} \tag{10}$$

where β is a learning rate, indicating the speed of the probability vector learned from the elite set $\mathcal{E}$. It seriously affects the convergence of APA. If β is close to 0, APA becomes a random sampling method. If β is too large, elements of the probability vector becomes constants quickly which traps the algorithm into local optimal.

For example, there are five elites in $\mathcal{E}$ with the resource renting costs $C = (20, 20, 25, 30, 40)$ and the cost of the global best solution $C(\pi^{\text{best}})$ being 15. According to Eq. (9), the weights of the elites are $w = (1.33, 1.33, 1, 0.67, 0)$. If the slack time ℓ_3 of task v_3 is 4, the possible shift states of v_3 would be $\{0, 1, 2, 3, 4\}$. Suppose the probability vector of v_3 is $\mathbf{p}_3 = (0.4, 0.4, 0.1, 0.05, 0.05)$ and the shift of task v_3 in each elite is $(0, 0, 1, 2, 3)$. The weighted number of states of v_3 in $\mathcal{E}$ is calculated by $\sum_{\pi_k \in \mathcal{E}} x_{jik} \cdot w(\pi_k)$, i.e., $(1.33 + 1.33, 1, 0.67, 0, 0) = (2.66, 1, 0.67, 0, 0)$ for each shift states. Assume the learning rate β is 0.5, the probability vector $\mathbf{p}_3$ of v_3 is updated by Eq. (10). For instance, $p_{30} = 0.5 \times 0.4 + 0.5 \times \frac{2.66}{2.66+1+0.67+0+0} = 0.507$. By this way, the new probability vector is obtained with $\mathbf{p}_3 = (0.507, 0.315, 0.127, 0.025, 0.025)$.

5 Computational experiments

To the best of our knowledge, there is no existing problem exactly the same as the considered problem. Workflow scheduling with hybrid resource provisioning is similar to the Resource Availability Cost Problem (RACP), a traditional single project scheduling problem [14]. For RACP, the typical meta-heuristics include the scatter

search (SS) [34], the genetic algorithm (GA) [28] and the path relinking algorithm (PR) [26]. The algorithm SS [34] adopted the resource capacity list to generate a schedule, while the algorithm GA [28] used the combination of an activity list and a resource capacity list to produce a schedule. They both transformed the problem into a series of resource-constrained scheduling problems and adjust the resource capacity to satisfy the deadline of the problem. The algorithm PR [26] generated a schedule directly according to the priority list of activities. In our algorithm APA, we use the shift vector scheme to generate a schedule directly. Comparing with APA and PR, SS and GA are much time-consuming. In addition, the three algorithms for RACP only consider reserved resources for scheduling workflows. For full evaluation, the three meta-heuristics are modified by adding IRPD to the schedule of each algorithm. The modified MSS, MGA and MPR are compared with the proposed APA. All algorithms are coded in Java and performed on the virtual machine (1000 MIPS processor with 1 G RAM and 10 GB of storage).

Different testing beds are adopted to calibrate the involved parameters and compare the algorithms. Problem sets with different sizes and deadlines are randomly generated by RanGen according to [31] to calibrate parameters. To compare the algorithms, project scheduling benchmarks from the well-known PSPLIB [21] are adapted for the considered problem. As well, RanGen is used to generate random instances with more type of resources and deadline factors DF to further compare the algorithms. Deadline factors DF are defined as $D = \mathrm{DF} \times est_n$, where est_n is the critical path of the current test workflow.

Since the schedule representation scheme and the computation of the improved heuristic are distinct in different methods, it is not impartial to use the number of iterations or CPU times as the terminal criterion. To fairly compare the involved algorithms, the maximum number of generated schedules is adopted in this paper. In MSS, every evaluated resource set is regarded as a schedule. In MGA and MPR, each evaluated feasible task sequence is regarded as a schedule. In the proposed APA, each evaluation of a shift vector is regarded as a schedule.

RDI (Relative Deviation Index) is adopted to measure the performance of the algorithms. Let π^{best} and π^{worst} be the best and worst schedules obtained by all the compared algorithms, respectively. π is the schedule obtained by the evaluated algorithm. RDI of an algorithm with resource renting cost $C(\pi)$ is defined as

$$RDI = \begin{cases} 0 & \text{if } \pi^{\mathrm{worst}} = \pi^{\mathrm{best}} \\ \frac{C(\pi) - C(\pi^{\mathrm{best}})}{C(\pi^{\mathrm{worst}}) - C(\pi^{\mathrm{best}})} \times 100\% & \text{otherwise} \end{cases}$$

The closer to 0 the index, the better the algorithm. Note that if the worst and the best solutions are the same, all the combinations would provide the best solution and hence the index value would be 0 (the best index value).

5.1 Parameter calibration

In the proposed APA, there are three parameters to be calibrated: the population size $|Pop|$, the elite size $|\mathcal{E}|$, the learning rate β, which are restricted to {50, 100, 150,

250}, {1, 3, 5, 7}, {0, 0.01, 0.03, 0.05}, respectively. The termination criterion (TC) takes value from {1000, 5000, 10000} according to [34]. Instances with 30, 60, 90 and 120 tasks (which are called as J30, J60, J90 and J120), and 4 types of resources are randomly generated by RanGen according to [31]. The order strength (OS), an index of the network complexity, is set as {0.25, 0.50, 0.75}. The resource factor (RF) indicates the density of the different resource types needed by a task equals to {0.25, 0.50, 0.75, 1}. For each combination of OS and RF, 5 instances are generated frp fair comparison. The deadline factor (DF) takes value from {1.1, 1.2, 1.3, 1.4, 1.5}, e.g., the deadline of the workflow application is defined as $D = 1.2 \times est_n \cdot c_k$ of resource R_k is a uniform random number in [1,10]. The discount φ is set to be 0.8 as an example. Totally, we test $4 \times 3 \times 4 \times 5 \times 5 = 1200$ different instances. For different combinations of parameters, there are $4 \times 4 \times 4 \times 3 \times 1200 = 230400$ tests.

The multifactor analysis of variance (ANOVA) technique is used to calibrate the three parameters. Let the response variable be RDI. First, the three main hypotheses (normality, homoscedasticity and independence of the residuals) are checked from the residuals. All three hypotheses are acceptable. Since all the p values in the experiments are close to zero, they are not given in this paper. Interactions between (or among) any two (or more than two) factors are not considered because the observed F-Ratios are small in comparison.

Interactions between $|Pop|$, $|\mathcal{E}|$ and task numbers with 95% Tukey HSD confidence intervals are depicted in Fig. 7. When $|Pop| = 100$, APA achieves the best performance for different problem size (n = 30, 60, 90 or 120). For the elite set size $|\mathcal{E}|$, except $J30$, the performances between different $|\mathcal{E}|$ are not so big. $|\mathcal{E}| = 3$ is the best one for all the problem size. The learning rate β exerts great influence on APA for different problem size, and APA obtains the best RDI when $\beta = 0.01$. Interactions between β, TC and task numbers with 95% Tukey HSD confidence intervals are depicted in Fig. 8. For the termination criterion TC, the performances improve a lot for all the problem size when TC increases from 1000 to 5000, which improves a little when TC increases from 5000 to 10000. In the following comparison, $|Pop|$ is

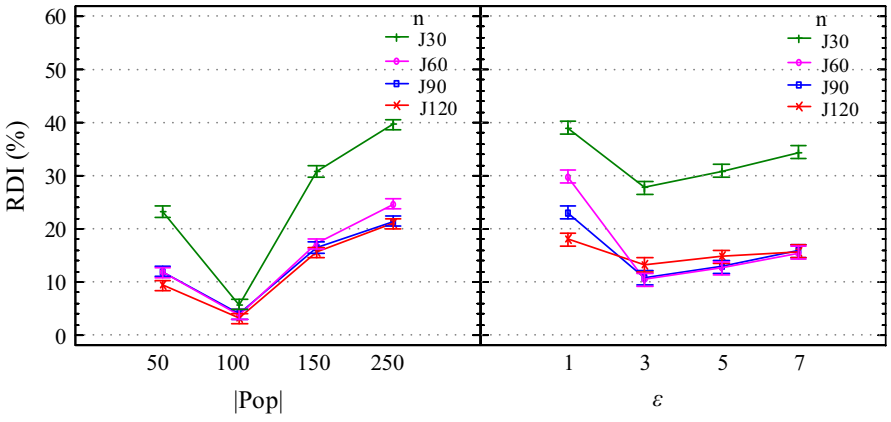


Fig. 7 Interactions between $|Pop|$, $|\mathcal{E}|$ and task numbers with 95% Tukey HSD confidence intervals on RDI

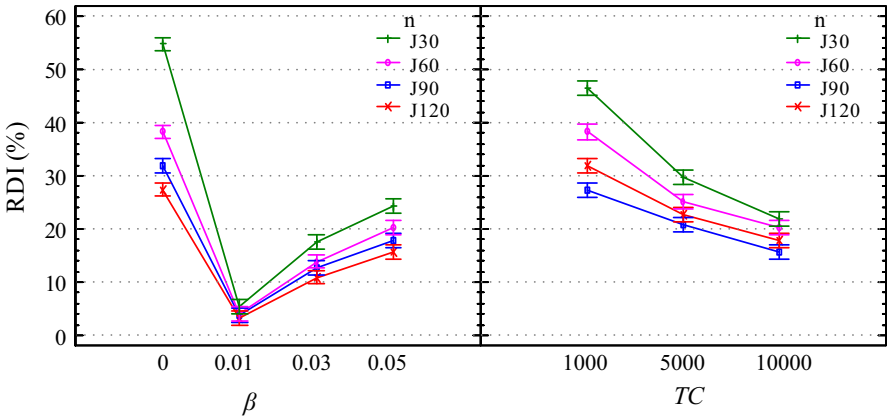
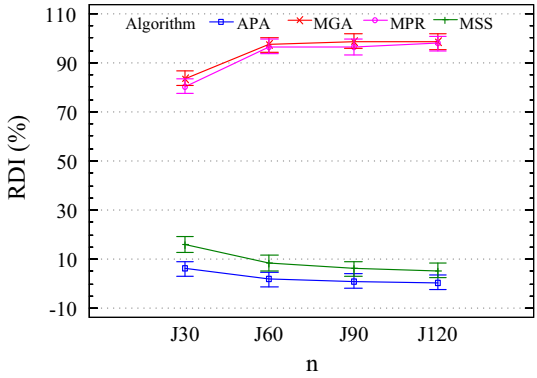


Fig. 8 Interactions between β, TC and task numbers with 95% Tukey HSD confidence intervals on RDI

Fig. 9 Interactions between algorithms and task numbers with 95% Tukey HSD confidence intervals on RDI



set at 100, $|\mathcal{E}| = 3$, $\beta = 0.01$ for APA, and the maximum number of schedules is set at 5000 for all the compared algorithms as it is the most common termination criterion for project scheduling problems [18].

5.2 Performance comparison on PSPLIB

Test beds with 30, 60, 90 and 120 tasks scheduling on one system with 4 types of resources chosen from PSPLIB are adapted for performance comparison of different meta-heuristics, in which NC $\in \{1.5, 1.8, 2.1\}$ reflects the average number of immediate successors of a task and RF $\in \{0.25, 0.5, 0.75, 1\}$ determines the average number of resources requested by a task. For each combination of NC and RF, 5 instances are selected. The deadline factor DF is set as 1.2 according to [34] and [26]. In total, $4 \times 1 \times 3 \times 4 \times 5 \times 1 = 240$ instances are taken from PSPLIB for the comparison.

Based on the calibrated parameters, RDI of the compared algorithms is shown in Fig. 9. From Fig. 9, it can be observed that the proposed APA outperforms the other three adapted algorithms on all four sizes of instances. For J30, the average RDI of

APA is close to 5%, while those of MSS, MGA and MPR are close to 15, 80 and 80%, respectively. The average RDI of APA and that of MSS decrease with the instance size, while this is not true for MGA and MPR. MGA and MPR get the worst average RDI on J90 and J120, whereas APA and MSS obtain the worst average RDI on J30 (about 5 and 15%, respectively), which implies that APA and MSS are more suitable for large instances than MGA and MPR.

The above experimental results over benchmarks reveal that APA gets a better performance comparing to the three adapted algorithms with the increase in the instance size. The reason lies in that the proposed SVT (Shift Vector-based Timetabling) scheme generates schedules directly. With the increase in the instance size, the adapted three algorithms require more steps and computation times to generate schedules as compared to APA.

5.3 Performance comparison on random sets

To further investigate the influence of resource types and deadlines on performances of the involved algorithms, instances with 30 tasks and 2, 4, 6, 8, 10 types of resources are randomly generated by RanGen. The deadline factor DF takes value from {1.1, 1.2, 1.3, 1.4, 1.5}. The order strength (OS) is set as {0.25, 0.50, 0.75}. And the resource factor (RF) belongs to {0.25, 0.50, 0.75, 1}. For each combination of OS and RF, 5 instances are generated. Totally, $3 \times 3 \times 4 \times 5 \times 5 = 900$ different instances are generated and tested.

Comparison results of all the test beds are shown in Table 2. Table 2 shows that the proposed APA outperforms the other three adapted algorithms in both effectiveness and efficiency on average. The average RDI of APA is only 2.15%, which is much less than those of the other three algorithms. APA spends just 15.70 s, while the average CPU time of MGA, MPR and MSS is 17.98.10, 16.33 and 33.84 s, respectively.

The means plot with 95% Tukey HSD confidence intervals for the compared algorithms is shown in Fig. 10. The average RDI of APA is about 5%, while that for MSS, MRP and MGA are about 45, 75 and 80%. Figure 10 indicates that APA can save more cost than MSS, MPR and MGA. Therefore, APA outperforms the other three methods both on benchmark and on random instances.

Interactions between algorithms and deadline factors with 95% Tukey HSD confidence intervals are shown in Fig. 11. Figure 11 illustrates that the proposed APA is better than the other three adapted algorithms on all the instances with different deadlines. Performance of MGA is the worst among the compared algorithms. The average RDI of APA is close to 0 while that of MGA is almost up to 80% for the test beds with different DF. The average RDIs of APA and MSS decrease with deadlines. However, this is not true for MGA and MPR. For example, APA saves about 55% of cost comparing with MGA, MPR and MSS when DF = 1.1, whereas the differences between APA and MSS, MPR, MGA become around 30, 80 and 80%, respectively, when DF = 1.5. The increase in the deadline factor leads to a better performance of APA. The reason is that a large deadline factor results in a large size of the shift probability distribution matrix and the diversification of APA is increased. However, APA takes more time to obtain schedules as the shift vector generation scheme (SVGS) is

Table 2 RDI (%) and CPU time (s) of the algorithms on random sets

DF	m	APA		MGA		MPR		MSS	
		RDI (%)	CPU (s)	RDI (%)	CPU (s)	RDI (%)	CPU (s)	RDI (%)	CPU (s)
1.1	2	**6.38**	2.78	39.95	**2.25**	36.45	2.50	25.78	5.32
	4	**2.33**	8.33	70.05	**6.74**	67.93	7.49	40.64	21.27
	6	**0.00**	13.17	90.98	**10.33**	90.17	10.88	88.60	28.08
	8	**9.27**	17.51	53.78	**14.29**	51.10	14.50	80.34	36.68
	10	**8.68**	24.51	51.66	**18.58**	48.31	18.85	76.13	47.68
1.2	2	**4.37**	**3.12**	48.88	3.26	48.49	3.18	14.01	5.63
	4	**0.05**	**9.36**	82.54	9.78	80.50	9.53	28.80	22.52
	6	**0.00**	14.75	95.02	13.47	92.93	**13.40**	76.33	29.80
	8	**0.13**	19.60	78.48	**15.89**	74.10	16.74	62.72	38.64
	10	**0.13**	27.44	74.57	**20.65**	70.83	21.77	59.48	57.96
1.3	2	**5.55**	**3.13**	50.67	3.53	46.19	4.00	13.29	5.92
	4	**0.19**	**9.39**	88.60	10.58	85.21	12.01	16.75	23.70
	6	**0.00**	**15.46**	96.27	17.38	93.87	17.35	67.47	31.18
	8	**2.06**	21.17	87.18	**19.88**	78.66	20.10	51.96	39.64
	10	**1.89**	29.64	83.28	**25.84**	75.07	26.13	49.43	71.35
1.4	2	**5.53**	**3.19**	48.01	4.50	55.54	4.66	11.72	6.00
	4	**0.29**	**9.57**	85.55	13.50	85.78	13.98	13.30	23.99
	6	**0.00**	**16.41**	98.31	21.02	96.66	18.59	60.15	31.96
	8	**0.00**	**22.40**	87.33	30.06	80.73	25.02	40.51	41.22
	10	**0.00**	**31.37**	83.36	39.08	76.82	32.53	38.48	82.43
1.5	2	**5.72**	**3.54**	48.35	6.15	44.35	5.53	11.10	6.20
	4	**0.07**	**10.62**	85.79	18.46	85.59	16.58	12.09	24.80
	6	**0.00**	**18.22**	99.19	24.16	97.02	22.27	53.77	33.01
	8	**0.57**	**24.10**	89.47	43.53	86.69	30.70	28.83	42.24
	10	**0.52**	**33.74**	85.26	56.58	82.65	39.91	27.40	88.71
Average		**2.15**	**15.70**	76.10	17.98	73.27	16.33	41.96	33.84

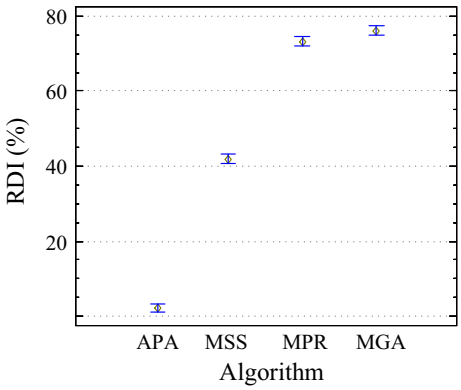


Fig. 10 Means plot with 95% Tukey HSD confidence intervals for the compared algorithms on RDI

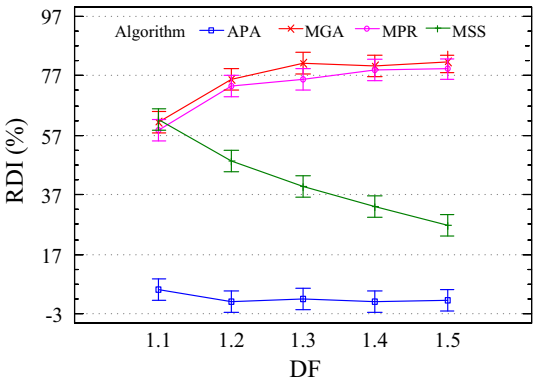


Fig. 11 Interactions between algorithms and deadline factors with 95% Tukey HSD confidence intervals on RDI (%)

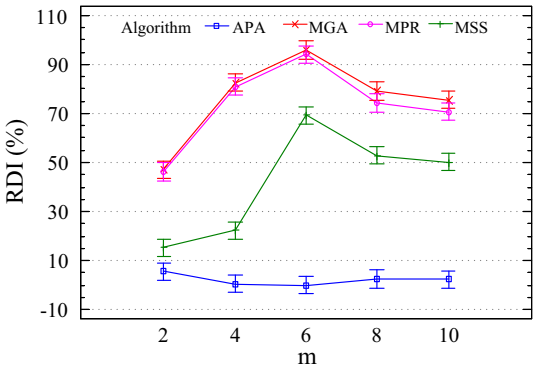


Fig. 12 Interactions between algorithms and resources types with 95% Tukey HSD confidence intervals on RDI (%)

time-interval based. With the increase in the deadline factor, the performance of APA would become worse because more time is required to find a solution in the search space.

Figure 12 demonstrates effectiveness of the compared algorithms with different numbers of resource types. APA is the best, while MGA is the worst for instances with different number of resource types. The average RDI of APA decreases with the number of resource types. However, the tendencies of MSS, MGA and MPR are non-monotonous. For example, APA saves about 5% of cost comparing with MSS, while it saves about 40% of cost comparing with MPR and MGA when $m = 2$. When $m = 6$, APA gets the best performance while the differences between APA and MSS, MPR, MGA become about 65, 85 and 87%, respectively. The above phenomena indicate that the traditional methods are less effective as the number of resource type increases in cloud computing. The reason lies in that the adapted algorithms generate schedules using resource capacity lists or priority lists of activities, and with the increase in the resource type, the adapted three algorithms also require more steps and computation times to generate schedules.

Finally, comparing results on test beds with different OS and RF are shown in Fig. 13. Figure 13 indicates that APA outperforms all the other algorithms. The average RDI of each algorithm has no monotonous trend when OS and RF increase.

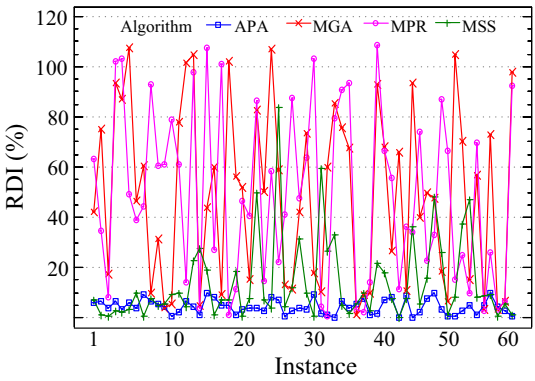


Fig. 13 Interactions between algorithms and instances on RDI (%) with 95% Tukey HSD confidence intervals for test beds with $m = 2$ and $DF = 1.2$

6 Conclusions and future work

We have considered the cloud workflow scheduling problem with hybrid resources provision which combination of the reserved and on-demand manners. The probability-based APA algorithm was proposed for the considered problem. APA mainly consisted of a new schedule generation scheme and an improvement procedure. The improvement procedure found good building blocks and the schedule generation scheme kept the good blocks to the next generation. Experimental results demonstrated that APA outperforms the other three adapted algorithms over both adapted benchmarks and random generated test beds. For the adapted benchmarks, APA improves MSS about 10%, MPR 85% and MGA 85% on average. For the random generated test beds, APA improves MSS about 40%, MPR about 75% and MGA 80% on average. With the deadline factor increased, the scale of the shift probability distribution matrix became larger which made APA more effective and efficient. Furthermore, the efficiency of APA decreased at a slower rate than the other algorithms as the number of resource types increased because the proposed schedule generation scheme produced timetables directly. The results indicated that APA is more suitable for resource allocation in cloud computing.

However, the proposed approach cannot be implemented in currently existent systems because many real constraints have not been considered in the problem under study, such as communication costs between tasks, setup times of virtual machines, uncertainties during executions. These constraints make the problem much harder. The proposed APA can be modified for those problems with newly designed heuristics incorporating characteristics of the harder problems. These topics would be promising in the future.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No. 61572127), the Key Research & Development program in Jiangsu Province (No. BE2015728) and Collaborative Innovation Center of Wireless Communications Technology.

References

1. Abrishami S, Naghibzadeh M, Epema D (2012) Cost-driven scheduling of grid workflows using partial critical paths. IEEE Trans Parallel Distrib Syst 23(8):1400–1414
2. Abrishami S, Naghibzadeh M, Epema DH (2013) Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener Comput Syst 29(1):158–169
3. AmazonEC2 (2016) Amazon elastic compute cloud (Amazon EC2). http://aws.amazon.com/ec2/pricing
4. Blythe J, Jain S, Deelman E, Gil Y, Vahi K, Mandal A, Kennedy K (2005). Task scheduling strategies for workflow-based applications in grids. In: IEEE International symposium on cluster computing and the grid (CCGrid 2005), vol 2. IEEE, pp 759–767
5. Byun EK, Kee YS, Kim JS, Deelman E, Maeng S (2011) BTS: resource capacity estimate for time-targeted science workflows. J Parallel Distrib Comput 71(6):848–862
6. Byun EK, Kee YS, Kim JS, Maeng S (2011) Cost optimized provisioning of elastic resources for application workflows. Future Gener Comput Syst 27(8):1011–1026
7. Cai Z, Li X, Gupta JND (2016) Heuristics for provisioning services to workflows in XaaS Clouds. IEEE Trans Serv Comput 9(2):250–263. doi:10.1109/TSC.2014.2361320
8. Cai Z, Li X, Gupta JND (2013) Critical path-based iterative heuristic for workflow scheduling in utility and cloud computing. In: Service-Oriented Computing. Springer, pp 207–221
9. Chaisiri S, Lee BS, Niyato D (2012) Optimization of resource provisioning cost in cloud computing. IEEE Trans Serv Comput 5(2):164–177
10. Chen Q, Wang L, Shang Z (2008) Mrgis: a mapreduce-enabled high performance workflow system for GIS. In: IEEE Fourth International Conference on eScience (eScience 2008). IEEE, pp 646–651
11. Chen WN, Zhang J (2009) An ant colony optimization approach to a grid workflow scheduling problem with various QoS requirements. IEEE Trans Syst Man Cybern C Appl Rev 39(1):29–43
12. Deelman E, Singh G, Su MH, Blythe J, Gil Y, Kesselman C, Mehta G, Vahi K, Berriman GB, Good J et al (2005) Pegasus: a framework for mapping complex scientific workflows onto distributed systems. Sci Program 13(3):219–237
13. Deelman E, Gannon D, Shields M, Taylor I (2009) Workflows and e-science: an overview of workflow system features and capabilities. Future Gener Comput Syst 25(5):528–540
14. Demeulemeester E (1995) Minimizing resource availability costs in time-limited project networks. Manag Sci 41:1590–1598
15. Demeulemeester E, Herroelen WS (2002) Project scheduling: a research handbook, vol 49. Kluwer Academic Publishers, Dordrecht
16. Demeulemeester E, Herroelen WS, Elmaghraby SE (1996) Optimal procedures for the discrete time/cost trade-off problem in project networks. Eur J Oper Res 88(1):50–68
17. Demeulemeester E, De Reyck B, Foubert B, Herroelen WS, Vanhoucke M (1998) New computational results on the discrete time/cost trade-off problem in project networks. J Oper Res Soc 49(11):1153–1163
18. Hartmann S, Kolisch R (2000) Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. Eur J Oper Res 127(2):394–407
19. Hazır Ö, Haouari M, Erel E (2010) Discrete time/cost trade-off problem: a decomposition-based solution algorithm for the budget version. Comput Oper Res 37(4):649–655
20. Juve G, Deelman E, Vahi K, Mehta G, Berriman B, Berman BP, Maechling P (2009) Scientific workflow applications on amazon EC2. In: 2009 5th IEEE International Conference on E-Science Workshops. IEEE, pp. 59–66
21. Kolisch R, Sprecher A (1997) Psplib-a project scheduling problem library: or software-orsep operations research software exchange program. Eur J Oper Res 96(1):205–216
22. Mazzucco M, Dumas M (2011) Reserved or on-demand instances? a revenue maximization model for cloud providers. In: IEEE International Conference on Cloud Computing (CLOUD), 2011. IEEE, pp. 428–435
23. Mohring RH (1984) Minimizing costs of resource requirements in project networks subject to a fixed completion time. Oper Res 32(1):89–120
24. Pelikan M, Goldberg D, Lobo F (2002) A survey of optimization by building and using probabilistic models. Comput Optim Appl 21(1):5–20
25. Radulescu A, Van Gemund AJ (2001) A low-cost approach towards mixed task and data parallel scheduling. In: International Conference on Parallel Processing (ICPP2001). IEEE, pp. 69–76

26. Ranjbar M, Kianfar F, Shadrokh S (2008) Solving the resource availability cost problem in project scheduling by path relinking and genetic algorithm. Appl Math Comput 196:879–888
27. Rodrigues SB, Yamashita DS (2010) An exact algorithm for minimizing resource availability costs in project scheduling. Eur J Oper Res 206:562–568
28. Shadrokh S, Kianfar F (2007) A genetic algorithm for resource investment project scheduling problem, tardiness permitted with penalty. Eur J Oper Res 181:86–101
29. Tang S, Yuan J, Wang C, Li XY (2014) A framework for amazon ec2 bidding strategy under sla constraints. IEEE Trans Parallel Distrib Syst 25(1):2–11
30. Topcuoglu H, Hariri S, Wu MY (2002) Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst 13(3):260–274
31. Vanhoucke M, Coelho J, Debels D, Maenhout B, Tavares LV (2008) An evaluation of the adequacy of project network generators with systematically sampled networks. Eur J Oper Res 187(2):511–524
32. Van den Bossche R, Vanmechelen K, Broeckhove J (2015) Iaas reserved contract procurement optimisation with load prediction. Future Gener Comput Syst 53:13–24
33. Wieczorek M, Prodan R, Fahringer T (2005) Scheduling of scientific workflows in the ASKALON grid environment. ACM SIGMOD Rec 34(3):56–62
34. Yamashita DS, Armentano VA, Laguna M (2006) Scatter search for project scheduling with resource availability cost. Eur J Oper Res 169:623–637
35. Yu J, Buyya R (2006) Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Sci Program 14(3):217–230
36. Yu J, Buyya R, Tham CK (2005) Cost-based scheduling of scientific workflow applications on utility grids. In: First International Conference on e-Science and Grid Computing (e-Science 2005). IEEE, p 8
37. Yuan Y, Li X, Wang Q, Zhu X (2009) Deadline division-based heuristic for cost optimization in workflow scheduling. Inf Sci 179(15):2562–2575