Cluster Computing (2024) 27:1401–1419
https://doi.org/10.1007/s10586-023-04022-w

Knowledge-based multi-objective estimation of distribution algorithm for solving reliability constrained cloud workflow scheduling

Ming Li[1] · Dechang Pi[1] · Shuo Qin[1]

Received: 25 January 2023 / Revised: 10 April 2023 / Accepted: 30 April 2023 / Published online: 14 May 2023

Abstract
With the rapid development of cloud computing, numerous large-scale workflow are executed in the cloud environment. Therefore, the workflow scheduling in cloud environment has become an emerging topic. This paper focuses on a reliability constrained multi-objective workflow scheduling problem (RCMOWSP) with the objectives of minimum execution cost and time. To solve the RCMOWSP, this paper proposes a knowledge-based multi-objective estimation of distribution algorithm (KMOEDA) with several problem-specific operators. First, an idle time-based decoding scheme is applied to sort the permutation of tasks greedily. In the global search strategy, a probability model is constructed to improve the diversity of population. Based on the problem-specific knowledge, a reliability-aware local search strategy is designed to performs local search around the solutions that violate reliability constraint. An elite enhancement strategy with a task perturbation operator and a resource perturbation operator is introduced to further improve the elite non-dominated solutions in the external archive. A comprehensive experiment is conducted to verify the performance of KMOEDA. The comparative results show that the KMOEDA significantly outperforms several relative multi-objective workflow scheduling approaches in solving the RCMOWSP.

Keywords Cloud computing · Workflow scheduling · Multi-objective optimization · Estimation of distribution algorithm · Reliability constraint

1 Introduction

Complex scientific computing programs are frequently described as workflows in several scientific domains, i.e., physics, biology, and astronomy [1, 2]. Directed acyclic graphs (DAGs) are applied to define the common workflow model [3]. In a DAG, the nodes represent computational tasks in a workflow, and the edges denote the relationships in data and control flow among various tasks. Due to the complex dependency relationships between tasks and the high amount of computation and transmission, it is a significant challenge to execute the workflow with reasonable time and cost [4].

As a prevalent computing paradigm, cloud computing can offer computing services on demand in a pay-as-you-go manner [5, 6]. Cloud service providers deploy numerous virtual machines on the servers to satisfy the requirements of widely dispersed users, which can effectively reduce the operational cost of services providers and the rental charges of users [7, 8]. Large-scale workflows are frequently deployed on the cloud computing platforms for execution because of the low cost, flexible application mode, and high availability. During the workflow execution process, the tasks must satisfy the dependency with other tasks and the available of virtual machines [9, 10]. Therefore, the tasks may be delayed because of the unsatisfied data and control dependencies or the occupied virtual machines, which increases total execution cost (*TEC*) and total execution time (*TET*). It is clear that the effective and efficient workflow scheduling algorithm can significantly reduce *TET* and *TEC* [11–13].

✉ Shuo Qin
qinshuo@nuaa.edu.cn

Ming Li
li_ming_nuaa@163.com

Dechang Pi
dc.pi@nuaa.edu.cn

[1] School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Shengtai West Road, Nanjing 211100, China

Existing workflow scheduling algorithms can be roughly divided into two categories, called real-time scheduling algorithms [14, 15] and predictive scheduling algorithms [16, 17]. Real-time scheduling algorithms focus on optimizing a single objective such as time and cost when meeting the desired constraints, i.e., budget and deadline. In addition, the real-time scheduling algorithms can fully utilize the transient resources in cloud environment when the resources requirement for tasks increase sharply. The computational complexity of a real-time scheduling algorithm is low, and then the real-time requirement can be satisfied. Predictive scheduling algorithms produce predictive schedules based on the estimated computational and transmission time. The running time of the predictive scheduling algorithm is long, and it is unable for real-time scheduling. The predictive scheduling algorithm obtains schedules and resources leasing scheme before workflow execution. Therefore, the predictive scheduling algorithms do not utilize transient resources. However, most of the real-time scheduling algorithms are heuristic approaches, which mainly depend on greedy search and are restricted by some predefined rules. Thus, the performance of the heuristic approaches is poor when addressing complex large-scale workflow scheduling problems. Therefore, the predictive scheduling algorithms, which are based on the intelligence optimization algorithms with strong global search technique, are superior to the real-time scheduling algorithms for the large-scale workflow scheduling problems [18–20].

Furthermore, multi-objective predictive workflow scheduling algorithms usually take numerous requirements of users into account and provide a set of trade-off schedules from which users can choose their favorite alternatives. This paper studies the multi-objective predictive workflow scheduling algorithm for the large-scale scientific workflow scheduling problem.

Most of the existing multi-objective predictive workflow scheduling algorithms focus on scheduling workflows with minimization *TET* and *TEC*. However, virtual machines in a cloud computing environment frequently experience momentary issues that prevent task execution [21]. To be specific, although some schedules achieves the low execution cost or fast execution time, their reliability may be extremely low. The low-reliability schedule results in a small probability to execute the workflow successfully. Additionally, the high reliability of the predictive schedules cannot guarantee the schedule seamless implementation, and requires expensive costs. Therefore, this paper focuses on the reliability-constrained multi-objective predictive workflow scheduling problem (RCMOWSP) in the cloud computing environment.

RCMOWSP involves three aspects, i.e., resource rental scheme, task allocation scheme, and task permutation. Moreover, the schedules should satisfy the predefined reliability constraints. However, there are too many schedules which do not meet the reliability constraint. Therefore, the existing algorithm cannot address this problem effectively because they will produce too many unfeasible solutions. In addition, the multi-objective cloud workflow scheduling problem typically requires high diversity of population, which can ensure the distribution of the final schedules. Thus, it is necessary to improve the global search ability of the algorithms. For these reasons, this paper proposes a knowledge-based multi-objective estimation of distribution algorithm (KMOEDA) to address the considered problem. KMOEDA apply the estimation of distribution algorithm (EDA), which is suitable for solving the large-scale optimization problem, to be the global search strategy. Besides, the core component of KMOEDA is the reliability-aware local search strategy, which can guide the search direction of algorithm and improve the search efficiency. The contributions of this paper can be summarized as follows.

- Formulate the reliability-constrained multi-objective workflow scheduling problem with the objectives of minimizing *TET* and *TEC*.
- A knowledge-based estimation of distribution algorithm with several problem-specific strategies is proposed to solve RCMOWSP.
- A reliability-aware local search strategy is designed to handle the schedules that violate the reliability constraint.
- The effectiveness and efficiency of KMOEDA are investigated by a set of experiments.

The remainder of this paper is structured as follows. Section 2 introduces the related work of the workflow scheduling algorithms in cloud environment. Section 3 describes the considered problem and related definitions. Section 4 details the proposed algorithm. Section 5 presents the parameter calibration, ablation analysis, and comparative experiments. In Sect. 6, the conclusion and several promising future directions are provided.

2 Literature review

The workflow scheduling algorithms consist of real-time workflow scheduling algorithms and predictive workflow scheduling algorithms. Efficiency is the most significant performance metric for real-time workflow scheduling algorithms. As a result, the real-time workflow scheduling problems can be effectively solved by the heuristic approach with low time complexity. The traditional heuristics for the real-time workflow scheduling problem are the HEFT algorithm [22], Max-Min algorithm [23], and CPOP algorithm [24]. Additionally, there are numerous heuristics for the workflow scheduling problems that take into account various factors, i.e., scalability [25], data localization [26], data transmission delay[27], bandwidth restriction [28], security[29], price[30], resource utilization [31], uncertainty [14], and fault tolerance [15].

Predictive workflow scheduling algorithms includes the deadline-constrained workflow scheduling algorithms, the budget-constrained workflow scheduling algorithms, and the multi-objective workflow scheduling algorithms. For the elastic and heterogeneous resources in cloud environment, a resource allocation and scheduling algorithm based on particle swarm optimization (PSO) was proposed by Rodriguez et al. [32] to address the deadline-constrained workflow scheduling problem with the objective of minimizing cost. Wang et al. [33] developed a distributed particle swarm optimization algorithm (DGLDPSO) based on the dynamic group learning strategy to minimize the execution cost with the deadline constraint for the large-scale workflow scheduling problem. Based on the virtual machines configurations in the actual public cloud and the execution data from the actual scientific workflow, Jia et al. [18] designed a novel task execution time estimation model and proposed an adaptive ant colony optimization algorithm (A-ACO) for workflow scheduling problem. To reduce the workflow execution cost under deadline constraint, Wu et al. [34] presented a meta-heuristic L-ACO and a heuristic ProLiS, where L-ACO is employed to optimize cost, and ProList is response for assigning the deadline constraint to each task. Faragardi et al. [17] suggested the resource supply and workflow scheduling algorithm GRP-HEFT to optimize the execution time when meeting the budget constraint. GRP-HEFT constructs the resource leasing scheme based on the resource execution efficiency using the greedy strategy, and applies the improved HEFT algorithm to schedule the tasks in the workflow.

To offer a set of trade-off Pareto solutions for users, Durillo et al. [35] developed a multi-objective HEFT algorithm (MOHEFT) to optimize the execution time and cost simultaneously. Then, users can select one of the schedules based on their preferences. However, MOHEFT requires an excessively high time and space complexity because it is an exhaustive-based approach. Therefore, MOHEFT is not appropriate for the large-scale workflow scheduling problem. Wu et al. [36] designed a multi-objective evolutionary list scheduling algorithm (MOELS) to address the multi-objective workflow scheduling problem. In MOELS, the schedule is encoded as the permutation in which tasks are executed, and the decoding procedure makes use of the HEFT algorithm. For resource management methods and payment strategies in the cloud computing environment, Zhu et al. [16] proposed a workflow scheduling strategy based on an evolutionary multi-objective optimization algorithm (EMS-C). Based on the problem-specific knowledge, EMS-C proposes a novel coding method, an initialization strategy, a fitness evaluation function, and genetic operators. According to the weight aggregation approach described in the literature [37], the execution cost and time are aggregated as an optimization objective. Then, the HEFT algorithm and gravitational search algorithm are combined to schedule the workflow. For optimizing the execution time, cost, and virtual machine energy efficiency, Paknejad et al. [38] developed an enhanced multi-objective co-evolution algorithm (ch-PICEA-g) based on chaotic systems. To minimize execution time, cost, and energy consumption and maximize reliability, an improved multi-objective particle swarm optimization algorithm (I_MaOPSO) is proposed by Saeedi et al. [39] for multi-objective workflow scheduling.

As above review, the classic multi-objective workflow scheduling problem with the objectives of minimizing cost and time has been successfully solved with the existing algorithms. However, the reliability of schedule has been ignored in the most of the previous researches. To be specific, the low reliability of schedule represents that the implementation of this schedule would fail with large probability. Then, the low execution cost and fast execution time of this schedule are meaningless. Moreover, a few existing approaches focus on maximizing the reliability when minimizing the time and cost. However, the high reliability often results in expensive cost and slow execution time. Considering the execution time and cost are the most concerned metrics of users, this paper tends to formulate a reliability-constrained multi-objective optimization problem to provide users with relatively reliable schedules. Then, the users can select one of the reliable schedules to execute the workflow on the cloud environment.

3 Problem statement

This section describes the considered problem from three aspects: the workflow model, the cloud resource model, and the reliability-constrained multi-objective cloud workflow scheduling model. The notations used in this section are displayed in Table 1.

3.1 Workflow model

Workflow model is denoted by the directed acyclic graph (DAG), $G = <T, E, W, D>$. $T = \{t_1, t_2, \cdots, t_n\}$ indicates the set of tasks. $E = \{(t_i, t_j)|t_i, t_j \in T\}$ denotes the dependency relationships between tasks. $W = \{w(t_1), w(t_2), \cdots, w(t_n)\}$ is the computational load of tasks. $D = \{d(t_i, t_j)|(t_i, t_j) \in E\}$ is the communication load between tasks. From above definitions, the predecessor

Table 1 The notations and descriptions

Notations	Descriptions
G	The given workflow
T	The set of tasks in G
E	The set of edges in G
W	The set of computational load of T
D	The set of communication load of E
t_i	The ith task of T
r_j	The jth resource of resources pool
n	The number of tasks
m	The number of resources
β	The bandwidth between two resources
u	The billing time unit of resources
δ_{Rel}	The predefined reliability constraint
p_i	The ith instance in cloud environment
$w(t_i)$	The computational load of task t_i
$\lambda(r_j)$	The failure rate of resource r_j
$d(t_i, t_j)$	The communication load between t_i and t_j
$rel(t_i, r_j)$	The reliability of t_i on resource r_j
$tr(t_p, t_i)$	The transmission time between t_p and t_i
$P(t_i)$	The set of precedence tasks of t_i
$C(t_i)$	The set of successor tasks of t_i
$Pr(r_j)$	The price of resource r_j for 1 billing unit
$CU(r_j)$	The processing capacity of resource r_j
$Cost(r_j)$	The rental cost of resource r_j
$ST(t_i)$	The start time of task t_i
$FT(t_i)$	The finish time of task t_i
$LST(r_j)$	The start time of resource r_j
$LET(r_j)$	The end time of resource r_j
$LDT(r_j)$	The consuming time of resource r_j
$ET(t_i, r_j)$	The execution time of t_i on r_j
$pri(t_i)$	The priority of task t_i
RS	The ready set of tasks
O	The execution permutation of tasks
$TS(r_j)$	The set of tasks assigned on resource r_j
R	The resource pool
MPT	The maximum number of parallel tasks
Rel	The total reliability for schedule
TET	The total execution time for schedule
TEC	The total execution cost for schedule

tasks set and the successor tasks set can be defined as $P(t_i) = \{t_p | (t_p, t_i) \in E\}$ and $C(t_i) = \{t_c | (t_i, t_c) \in E\}$. Figure 1 depicts a workflow model made up of 10 tasks.

From Fig. 1, it can be observed that the predecessor tasks set of task t_5 is $P(t_5) = \{t_1, t_4\}$, the successor tasks set of task t_5 is $C(t_5) = \{t_7, t_9\}$. The computational load $w(t_5)$ is 5, and the communication load $d(t_5, t_7)$ is 4.

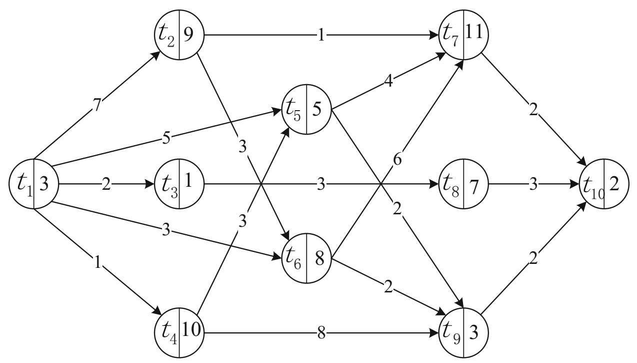


Fig. 1 A workflow model with 10 tasks

3.2 Cloud resource model

Cloud computing provides computational services in the form of virtual machines. The users can rent any number of any kind of heterogeneous virtual machines at any time. To be specific, there is an enormous number of virtual machine configurations in a cloud environment, but the cost to rent a virtual machine increases with computational efficiency. Moreover, the users must pay for a full-time unit even if the virtual machine is used for a short time. For any resource r_j, $LST(r_j)$ represents the start time, $LET(r_j)$ denotes the end time, and $LDT(r_j)$ means the holding time. Let $Pr(r_j)$ represents the unit rental price of r_j, the total rent cost of resource r_j can be calculated as follows.

$$Cost(r_j) = Pr(r_j) \times \lceil LDT(r_j)/u \rceil \tag{1}$$

where u is the rent time unit. In this paper, the hourly billing mechanism is used, which means u is set to $1h$.

3.3 Reliability-constrained multi-objective workflow scheduling model

We assume that task t_i is deployed on resource r_j, and the processing capacity of r_j is $CU(r_j)$, the execution time $ET(t_i, r_j)$ of task t_i on resource r_j is defined as follow.

$$ET(t_i, r_j) = w(t_i)/CU(r_j) \tag{2}$$

Furthermore, the data transmission time $tr(t_p, t_i)$ from any predecessor task t_p to task t_i can be calculated as below.

$$tr(t_p, t_i) = \begin{cases} \dfrac{d(t_p, t_i)}{\beta}, & \text{if } r(t_p) \neq r(t_i) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $r(t_p)$ and $r(t_i)$ are the resources which execute task t_p and t_i. β is the bandwidth between resources $r(t_p)$ and $r(t_i)$. Furthermore, The start time of task t_i is constrained by two factors, (1) the task must delay until all of the communication data from its immediate predecessor tasks has been received; (2) the task can be executed when the corresponding resource is free. As a result, the task start time $ST(t_i)$ can be calculated as follow.

$$ST(t_i) = max\{max\{FT(t_p) + tr(t_p, t_i)\}, LET(r(t_i))\}, \quad t_p \in P(t_i) \tag{4}$$

where $FT(t_p)$ denotes the finish time of task t_p, $max\{FT(t_p) + tr(t_p, t_i)\}$ indicates that the task t_i have received all communication data from its predecessor tasks. $LET(r(t_i))$ is the finish time of the previous task on the same resource. Then, the finish time $FT(t_i)$ can be calculated as follow.

$$FT(t_i) = ST(t_i) + ET(t_i) \tag{5}$$

Furthermore, the resource start time $LST(r_j)$ is equal to the start time of the first task executed on r_j, and the resource end time $LET(r_j)$ is equal to the finish time of the last task executed on r_j.

$$LST(r_j) = min\{ST(t_i) | r(t_i) = r_j\} \tag{6}$$

$$LET(r_j) = max\{FT(t_i) | r(t_i) = r_j\} \tag{7}$$

$$LDT(r_j) = LET(r_j) - LST(r_j) \tag{8}$$

Since RCMOWSP is a predictive scheduling problem which generates schedules before the workflow is executed, there is no scheduling consuming in the workflow execution process. Then, the total execution time (*TET*) and total execution time (*TEC*) can be calculated as follows.

$$TET = max\{ET(t_i | t_i \in T)\} \tag{9}$$

$$TEC = \sum_{j=1}^{m} Cost(r_j) \tag{10}$$

In the cloud environment, the transient failure caused by virtual machine crashes and software defects will interrupt the workflow execution, and then the schedules cannot be implemented as expected. In this paper, we assume that the failure of virtual machines follows Poisson distribution. Let $\lambda(r_j)$ denotes the failure rate of resource r_j, the reliability of task t_i on resource r_j can be defined as follow.

$$rel(t_i, r_j) = e^{-\lambda(r_j) \times ET(t_i, r_j)} \tag{11}$$

Then, the reliability of the whole workflow can be calculated as

$$Rel = \prod_{i=1}^{n} rel(t_i, r(t_i)) \tag{12}$$

Therefore, the optimization objective of the considered problem is to minimize *TET* and *TEC*. δ_{Rel} is a user-defined reliability threshold, which means that a feasible schedule must meet the constraint $Rel > \delta_{Rel}$.

4 Proposed algorithm

Due to the RCMOWSP not being solved by the previous research, this paper proposes a knowledge-based multi-objective estimation of distribution algorithm (KMOEDA) to solve the considered problem. Considering the elastic and heterogeneous resources, the resource pool is constructed based on the maximum number of parallel tasks in KMOEDA. A decoding technique focused on idle time is designed throughout the decoding process to greedily sort the order of the tasks on the same virtual machine. Based on the above strategies, the problem space of RCMOWSP is reduced to the mapping relationships between tasks and virtual machines. Then, an effective initial strategy is designed to construct promising solutions. To prevent premature convergence and ensure the diversity of the population, a global search strategy is designed based on the estimation of distribution algorithm. Estimation of distribution algorithm can be classified as univariate, bivariate, and mutivariate due to variations in statistics. As the probability model in KMOEDA is a univariate model, the proposed KMOEDA is univariate. In addition, a reliability-aware local search strategy is proposed to handle the solutions that violate reliability constraints. Furthermore, an elite enhancement strategy for the elite solution in the external archive is developed based on the critical task and resource perturbation operators. The details of all components are explained below.

4.1 Encoding and decoding strategy

Because of the elasticity and Heterogeneity of virtual machines, the construction of a resource pool has a significant impact on the schedules. Due to each task in the workflow is only executed once, and the task can only be executed by one resource at the same time, a workflow of size n can occupy up to n resources. Obviously, the maximum size of the resource pool is n. In the previous research, there are two popular strategies for constructing the resource pools during the optimization process. The first one is to set the maximum size of the resource pool to n, and the type of each resource is dynamically optimized during the search process, which is the dynamic resource pool [16]. The second one is to preset the resource pool based on the maximum number of parallel tasks, and the resource type remains unchanged during the search process, called static resource pool [40]. Compared to the dynamic resource pool, the static resource pool can effectively reduce the complexity of problem space. Since only one resource can execute a task at a time, the following problem-specific knowledge can be discovered. For a given workflow with n tasks, the maximum number of leasing virtual machines is n. We assume that the cloud environment provides k configurations of virtual machines. Then, the maximum number of virtual machines in the resources pool is $n \times k$. However, not all tasks can be executed at the same time because of the topology constraints between tasks. Therefore, we construct a resource pool by using the maximum number of parallel resources. We assume that the given contains n tasks, and each resource only executes one task. Then, the maximum number of parallel resources MPT can be calculated based on the start time and finish time of each task. Therefore, the number of virtual machines in the resources pool is $MPT \times k$, where the instance of $\{r_1, r_2, \cdots, r_{MPT}\}$ is p_1, the instance of $\{r_{MPT+1}, r_{MPT+2}, \cdots, r_{2\times MPT}\}$ is p_2, and so on.

Figure 2 depicts the encoding strategy that the workflow in Fig. 1 is scheduled to the resources pool R. In Fig. 2, the first row demonstrates the tasks in the workflow, and the second row indicates the deployment of tasks on resources. For example, the first position of the second row is 1, which means that the task t_1 is assigned to the resource with index 1. For any task, the start time is limited by the resource release time and the time that all direct predecessor tasks transfer data are received. When the time of receiving data is later than the resource releasing time, the idle time slot will appear. Because the resources are leased on the hourly billing mechanism, using the idle time slots can effectively reduce the execution time without raising the resource leasing cost. Therefore, this paper designs an idle time-based decoding strategy to construct the execution order of tasks. The decoding strategy consists of two phases. In the first phase, the tasks are ordered according to the average execution time and data transmission time of tasks. The task priority can be calculated as follow.

t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
1	1	4	7	6	3	2	4	6	2

Fig. 2 An example of the encoding strategy

$$pri(t_i) = \overline{ET(t_i)} + \max_{t_c \in C(t_i)} \left(\overline{tr(t_i, t_c)} + pri(t_c)\right) \tag{13}$$

where $\overline{ET(t_i)}$ is the average execution time of t_i on all available instances, and $\overline{tr(t_i, t_c)}$ is the average transmission time between t_i and t_c. In the second phase, the tasks without predecessors in T are firstly added to the ready tasks set RS, and the task permutation O is an empty set. Then, the task with the highest priority is chosen and the earliest start time $EST(t_k)$ is calculated. When $EST(t_k)$ is larger than $LET(r(t_k))$, an idle time slot will appear on $r(t_k)$ if the task t_k is directly executed. Therefore, all ready tasks which are deployed on $r(t_k)$ are carried out to construct a set TS in the order of task priority. Then, the earliest finish time $EFT(t_i)$ of each task is calculated. When $EFT(t_i)$ is smaller than $EST(t_k)$, the task t_i can be executed in the idle time slot. Therefore, t_i is appended to the tail of O. If the earliest finish time of all tasks is larger than $EST(t_k)$, t_k is directly appended to the tail of O. Next, the edges between t_k and other tasks are removed, and the tasks in T without predecessors are added to RS. The above procedure is repeated until RS is an empty set. The decoding strategy is described in Algorithm 1.

Algorithm 1 Decoding strategy

```
1:  Sort the tasks according to task priority
2:  Init ready tasks set $RS \leftarrow \{t_i|P(t_i) = \emptyset\} \cap T$, $O \leftarrow \emptyset$
3:  while $RS \neq \emptyset$ do
4:      $t_k \leftarrow argmax(pri)$, Calculate $EST(t_k)$
5:      if $EST(t_k) \geq LET(r(t_k))$ then
6:          $TS \leftarrow \{t_i|r(t_i) \wedge t_i \in RS\}$
7:          Sort $TS$ according to task priority
8:          for $t_i \in TS$ do
9:              Calculate $EFT(t_i)$
10:             if $EFT(t_i) < EST(t_k)$ then
11:                 $t_k \leftarrow t_i$, break
12:             end if
13:         end for
14:     end if
15:     Append $t_k$ to the tail of $O$
16:     for $t_i \in C(t_k)$ do
17:         Remove edge $(t_k, t_i)$
18:         if $P(t_i) = \emptyset$ then
19:             $RS = RS \cup \{t_i\}$
20:         end if
21:     end for
22: end while
```

Suppose that the processing capacity of all resources are 1 and the bandwidth between resources are 1, the priority of tasks in the workflow shown in Fig. 1 is $\{48, 41, 16, 37, 24, 29, 15, 12, 7, 2\}$. Therefore, the task t_1 with the largest priority is chosen. Due to task t_1 has no predecessors and the assigned resource r_1 can be used immediately, the task t_1 is executed on the first position of resource r_1, $FT(t_1)$ and $LET(r_1)$ are 3. Then, the task t_2 has the largest priority among the residual tasks, and the $EST(t_2)$ is 3 because t_1 is the predecessor of t_2. As $EST(t_2) \geq LET(r_1)$, the remaining tasks assigned to r_1 form a set $TS = \{t_2\}$. Since there are no other tasks on r_1, the task r_2 is directly assigned to the second position of r_1. Thus, $FT(t_2)$ is 12, $LET(r_1)$ is updated to 12, and t_4 becomes the task with the largest priority. As the resource r_7 can be used immediately, the start time of t_4 is 4 because it can receive the data from its predecessor t_1 at time 4. Then, $FT(t_4)$ is 14, $LET(r_7)$ is 14, and the task t_3 becomes the task with the largest priority. The above procedure is repeated until all tasks have been assigned.

4.2 Global search strategy

The estimation of distribution algorithm (EDA) [41] constructs and updates the probability model by sampling the elite solutions in the population, and then the trial population is generated based on the probability model. The conventional EDA consists of the following steps: (1) Construct the probability model; (2) Generate the new population based on the probability model; (3) Evaluate the solutions in the new population; (4) Update the probability model based on the elite solutions. The above steps are repeated until the termination condition is satisfied.

4.2.1 Construction of probability model

Based on the above encoding and decoding scheme, this paper only considers the mapping relationships between tasks and virtual machines. Therefore, this paper constructs a univariate probability model P to reflect the probability that each task t_i is deployed to each resource r_j. We assume that the number of tasks is n and the number of virtual machines is m, $p_{i,j}$ represents the probability that task t_i is deployed to the resource r_j. Due to P is constructed based on the average probability of tasks being assigned to a resource among multiple candidate solutions, and the average probability is a first-order statistic, the probability model P is a univariate probability model. The probability model is shown as follows.

$$P = \begin{bmatrix} p_{1,1} & p_{2,1} & \cdots & p_{n,1} \\ p_{1,2} & p_{2,2} & \cdots & p_{n,2} \\ \vdots & \vdots & \ddots & \vdots \\ p_{1,m} & p_{2,m} & \cdots & p_{n,m} \end{bmatrix} \tag{14}$$

In the initial phase, the probability $p_{i,j}$ is set to $1/m$.

4.2.2 Update strategy of probability model

In KMOEDA, the external archive $\mathbb{L}$ is carried out to store the found non-dominated solutions. Therefore, the probability model is updated based on the elite solutions in $\mathbb{L}$. To make the probability model balance the historical information and the information of current elite solutions, the population-based incremental learning technique (PBIL) is applied for updating the probability model as shown in Eq. 15.

$$p_{i,j}(t+1) = (1-\alpha) \times p_{i,j}(t) + \alpha \times I_{i,j}(t)/|\mathbb{L}| \tag{15}$$

where $\alpha \in [0, 1]$ denotes the learning rate, $I_{i,j}(t)$ represents the frequency that task t_i is deployed to resource r_j. According to the previous literature [42], the PBIL is also a univariate method.

4.2.3 Construction of trial population

One of the essential elements of EDA is the creation of the trial population. Although the mapping relationships between tasks and resources can be well reflected by the probability model, solutions obtained by direct sampling will lose the structural information between tasks. Therefore, this paper introduces the reference template-based sampling approach [43] to construct the trial population. A random solution π_i is first chosen from the external archive $\mathbb{L}$, and then $d \times n$ tasks mapping are randomly eliminated from s_i. Next, the removed tasks are re-deployed on the virtual machines in the resources pool using the probability model P and roulette wheel selection technique. The procedure of global search strategy is described in Algorithm 1.

Algorithm 2 Global search strategy

```
1: for $\pi_k \in \mathbb{L}$ do
2:     $P(t+1) \leftarrow$Update $P(t+1)$ according to $\pi_k$ and $P(t)$
3: end for
4: for $\pi_k \in TP$ do
5:     $\pi_{temp} \leftarrow$Choose reference template from $\mathbb{L}$
6:     $\sigma \leftarrow$ Move $d \times n$ task from $\pi_{temp}$ to $\sigma$.
7:     $\pi_k \leftarrow \pi_{temp}$
8:     for $t_i \in \sigma$ do
9:         $r_{new} \leftarrow$ Choose resource for $t_i$ according to $P(t+1)$
10:        $\pi_k(t_i) \leftarrow r_{new}$
11:    end for
12: end for
```

For example, suppose that m is 7, n is 10, α is 0.5, d is 0.2, and there are two solutions $s_1 = \{1, 1, 4, 7, 6, 3, 2, 4, 6, 2\}$ and $s_2 = \{1, 2, 6, 3, 4, 5, 2, 7, 1, 4\}$ in the external archive $\mathbb{L}$, each element of the probability model P in the initial phase is $\frac{1}{7}$. The matrix I can be constructed as follows.

$$I = \begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{16}$$

Then, the probability model P can be updated as follows.

$$P = \begin{bmatrix} \frac{4}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} \\ \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{4}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} \\ \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{9}{28} \\ \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} \\ \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} \\ \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} & \frac{1}{7} & \frac{9}{28} & \frac{1}{7} & \frac{1}{7} \end{bmatrix} \tag{17}$$

Thus, the solution s_1 is chosen from $\mathbb{L}$ to be reference template, and 3 tasks $\sigma = \{t_3, t_5, t_7\}$ are randomly removed from s_1. Then, based on the roulette wheel selection technique, t_3 is re-deployed to r_6, t_5 is re-deployed to r_3, and t_7 is re-deployed to r_4. Thus, a new solution is generated. The above procedure should be performed on all solutions in the trial population.

4.3 Reliability-aware local search strategy

Since the reliability of task follows the Poison distribution, it can be concluded that the reliability of task is only related to the failure rate of virtual machines and the execution time of task. Then, the total reliability can be formulated as follow.

$$Rel = \prod_{i=1}^{n} rel(t_i) = e^{-\sum_{i=1}^{n} \lambda(r_(t_i))ET(t_i, r(t_i))} \tag{18}$$

Suppose that $Rel \geq \delta_{Rel}$, then

$$\sum_{i=1}^{n} \lambda(r_(t_i))ET(t_i, r(t_i)) \leq -log(\delta_{Rel}) \tag{19}$$

Therefore, the reliability constraint can be converted to a weighted execution time constraint. Since the failure rate of virtual machines is constant in a specified period, the reliability of a task decreases as the execution time increases. Therefore, a reliability-aware local search strategy is designed to search for feasible solutions near infeasible ones. The procedure of the local search strategy is described in Algorithm 3. First, the reliability constraint is converted to a weighted execution time constraint, and then the weighted execution time constraint is assigned to each task. The weighted execution time constraint $\delta_{wt(t_i)}$ of task t_i can be calculated as follow.

$$\delta_{wt(t_i)} = -w(t_i) / \sum_{i=1}^{n} w(t_i) \times log(\delta_{Rel}) \tag{20}$$

Second, the task list $TS(r_j)$ of each resource r_j is constructed. Then, the total weighted execution time $wt(r_j)$ of each resource and the total weighted execution time constraint $\delta_{wt(r_j)}$ are obtained as follow.

$$wt(r_j) = \sum_{i=1}^{|TS|} wt(t_i), \ TS = \{t_i | r(t_i) = r_j\} \tag{21}$$

$$\delta_{wt(r_j)} = \sum_{i=1}^{|TS|} \delta_{wt(t_i)}, \; TS = \{t_i | r(t_i) = r_j\} \tag{22}$$

Then, the resources are sorted according to $\Delta wt(r_j) = \delta_{wt(r_j)} - wt(r_j)$. The resources with maximum $\Delta wt(r_j)$ are iteratively chosen to modify the instance until the weighted execution time satisfies the constraint. The procedure of the local search strategy is shown in Algorithm 3.

Algorithm 3 Reliability-aware local search strategy

```
1:  Calculate $\delta_{wt(t_i)}$ of each task
2:  for $t_i \in T$ do
3:      Add $t_i$ into $TS(r(t_i))$
4:  end for
5:  for $r_j \in R$ do
6:      Calculate $\Delta wt(r_j)$ of each resource
7:  end for
8:  Sort resources according to $\Delta wt(r_j)$
9:  for $r_j \in R$ do
10:     for $p_k \in P$ do
11:         Calculate $\Delta wt(r_j)$ when $p(r_j) = p_k$
12:         if $\Delta wt(r_j) \leq 0$ then
13:             Move $TS(r_j)$ from $r_j$ to another resource with $p_k$
14:             Break
15:         end if
16:     end for
17:     if $Rel \geq \delta_{Rel}$ then
18:         Break
19:     end if
20: end for
```

Suppose that the user-defined reliability threshold δ_{Rel} is 50%, and the failure rate λ of the resources are {0.1,0.07,0.03,0.02,0.008,0.006,0.001}. According to Eq. 20, the weighted execution time constraints of tasks are {0.0352, 0.1057, 0.0117, 0.1175, 0.0587, 0.0940, 0.1292, 0.0822, 0.0352, 0.0235}. For solution s_1 shown in section 4.2.3, the actual weighted execution time of the tasks are {0.3, 0.9, 0.02, 0.01, 0.03, 0.24, 0.77, 0.14, 0.018, 0.14}. Furthermore, the weighted execution times of resources and the weighted execution time constraint are {1.2, 0.91, 0.24, 0.16, 0, 0.048, 0.01} and {0.1409, 0.1527, 0.094, 0.0939, 0, 0.0939, 0.1175}. Then, it can be obtained that $\Delta_{wt(r_j)}$={−1.0591, −0.7573, −0.1460, −0.0661, 0, 0.0459, 0.1075}. Due to $\Delta_{wt(r_1)}$ is the smallest, the r_1 is firstly chosen to change the instance of the resource. For the tasks t_1 and t_2 on resource r_1, it can be calculated that the weighted execution times on all instances are {1.2000, 0.8400, 0.3600, 0.2400, 0.0960, 0.0720, 0.0120}. Obviously, when the instance of r_1 is p_5, $\Delta_{wt(r_1)}$ is larger than zero. Therefore, the tasks t_1 and t_2 are moved from r_1 to another resource with the instance of p_5. Then, the reliability of the schedule is calculated again. This procedure is repeated until the reliability of the schedule meets the user-defined reliability constraint.

4.4 Elite enhancement strategy

To further explore the problem space deeply, this paper applies an elite enhancement strategy to improve the quality of non-dominated solutions in $\mathbb{L}$. The elite enhancement strategy consists of two perturbation operators, i.e., task perturbation and resource perturbation.

- **Task perturbation** first calculates the critical path of the solution, and then a random task on the critical path is moved to a random resource. This operator can effectively disturb the total execution time with constant total execution cost.
- **Resource perturbation** first chooses a random resource on which at least one task is deployed, and then move all tasks deployed on this resource to another virtual machine. This operator can effectively disturb the total execution cost.

In each iteration, two random solutions in L are carried out to perform the elite enhancement strategy. The task perturbation is performed on the first solution, and the resource perturbation is executed on the second solution. Suppose that the solutions s_1 and s_2 shown in section 4.2.3 are chosen to perform the elite enhancement strategy. The critical path of s_1 is $\{t_1, t_2, t_6, t_7, t_10\}$, and a random task t_6

is moved from r_3 to r_2. Therefore, the data communication between t_6 and t_7 becomes zero, and the total execution time of s_1 is reduced. Then, it can be observed that there are seven resources used in s_2. First, a random resource r_2 is chosen. The tasks assigned to r_2 are $\{t_2, t_7\}$. Second, tasks t_2 and t_7 are moved from r_2 to a random resource r_7. Thus, there is no task on r_2.

4.5 External archive

The output of the KMOEDA algorithm is a set of non-dominated solutions, that is, there is no solution that the execution time, cost, and reliability are superior to the other solutions at the same time. Therefore, the external archive $\mathbb{L}$ is applied to preserve the non-dominated solutions found during the iterations.

4.5.1 Initialization of external archive

Numerous previous studies have demonstrated that solid initial solutions can significantly accelerate the convergence speed and increase the quality of solution [44, 45]. Therefore, this paper applies two heuristics to initialize the external archive. First, all tasks are deployed to the same resource for execution. For k available instances, we can obtain k init solutions. Second, a solution is constructed based on the popular HEFT algorithm. Therefore, the initial external archive contains $k + 1$ solutions.

4.5.2 Update strategy of external archive

In each iteration, we can obtain $ps + 2$ solutions after the global search strategy and the local search strategy. First, the trial population TP and the external archive $\mathbb{L}$ are emerged, then the non-dominated fast sorting technique [46] is performed to generate the non-dominated solutions set. When the number of solutions in the non-dominated solutions set is larger than rs, the crowding distance [46] is employed to sort the solutions, and then the rs solutions with larger distance are remained in external archive $\mathbb{L}$.

4.6 Framework of KMOEDA

Based on the above components, this paper proposes a knowledge-based multi-objective distributed estimation algorithm (KMOEDA). The flowchart of KMOEDA is shown in Fig. 3. First, the external archive is constructed based on the heuristics, and the probability model is initialized. Second, the mapping relationship between tasks and virtual machines in the external archive is counted, and then the probability model is updated. Based on the probability model and the reference templates, the trial population is constructed. Moreover, the elite enhancement strategy is performed to generate two improved elite solutions. For the solutions which violate the reliability in the trial population, the reliability-aware local search strategy is applied to search for feasible solutions. Next, the external archive is updated. The above procedure is repeated until the termination criterion is met.

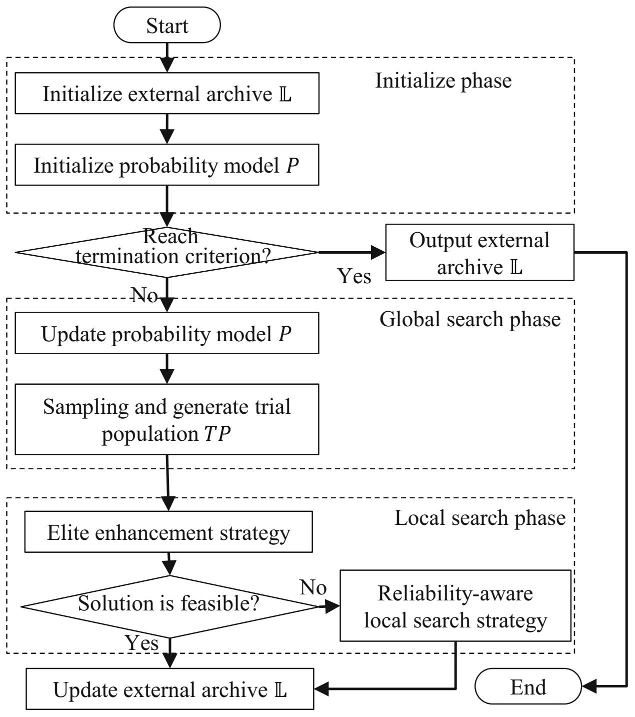

Start
Initialize phase
Initialize external archive $\mathbb{L}$
Initialize probability model P
Reach termination criterion?
Yes
No
Output external archive $\mathbb{L}$
Global search phase
Update probability model P
Sampling and generate trial population TP
Local search phase
Elite enhancement strategy
Solution is feasible?
No
Yes
Reliability-aware local search strategy
Update external archive $\mathbb{L}$
End

Fig. 3 The flowchart of KMOEDA

4.7 Computational complexity

Suppose that the workflow G is made up of n tasks, the resources pool contains m virtual machines, and there are p instances in the cloud environment. Let s denotes the size of the external archive, and k represents the size of the trial population. In the initialize phase, the computational complexity of the init external archive is $O(pn + mn^2)$, and the computational complexity of the init probability model is $O(mn)$. Therefore, the total computational complexity of initializing phase is $O(pn + mn^2 + mn)$. In each iteration, the computational complexity of updating the probability model is $O(sn)$, and the computational complexity of constructing the trial population is $O(kmn)$. In the elite enhancement strategy, the computational complexity of searching critical path is $O(n)$. Therefore, the computational complexity of the elite enhancement strategy is

$O(n+2)$. The computational complexity of the reliability-aware local search strategy is $O(pn)$, and the computational complexity of the decoding strategy is $O(n^2)$. The computational complexity of the fitness function is $O(n^2)$, and the computational complexity of the non-dominated sorting approach is $O(2s^2)$. Therefore, the computational complexity of KMOEDA in g iterations is as follows.

$$\begin{aligned} O(W) = {} & O(pn + mn^2 + mn) \\ & + O(g(sn + mn + n + 2)) \\ & + O(g((k+2)(pn + 2n^2) + 2s^2)) \end{aligned} \tag{23}$$

In the worst case, the number of virtual machines m is equal to pn. For the large-scale workflows, n is over larger than p. Therefore, the computational complexity of KMOEDA is

$$\begin{aligned} O(W) = {} & O(pn + pn^3 + pn^2) \\ & + O(g(sn + pn^2 + n + 2)) \\ & + O(g((k+2)(pn + 2n^2) + 2s^2)) \\ & \approx O(pn^2 + gpn^2 + (2k+4)gn^2) \\ & \approx O((2k+p+4)gn^2) \end{aligned} \tag{24}$$

Table 2 Configurations of the instances

Instances	CU	β	λ	Price
m4.large	1000	20	0.1	0.16
m4.xlarge	2000	20	0.07	0.26
m4.2xlarge	4000	20	0.03	0.53
m4.4xlarge	6000	20	0.02	0.93
m4.10xlarge	8000	20	0.008	2.13
m4.16xlarge	10000	20	0.006	3.33

5 Experimental study

5.1 Experimental setup

To evaluate the effectiveness of the proposed KMOEDA, we compared the proposed KMOEDA with several related algorithms in the same experimental environment. The configurations of instances in the experiment is simulated with reference to six virtual machines on Amazon EC2 cloud platform. The billing time unit is set to $1h$. The details of the configurations of virtual machines are described in Table 2.

Four scientific workflows, i.e., CyberShake, Montage, Inspiral, and Sipht, were employed in this experiment as workflow cases. These workflows are frequently utilized in the assessment of workflow scheduling algorithms [47]. Figure 4 depicts the topology structures of these workflows. The workflowSim is utilized to simulate the scheduling environment in the cloud environment. To

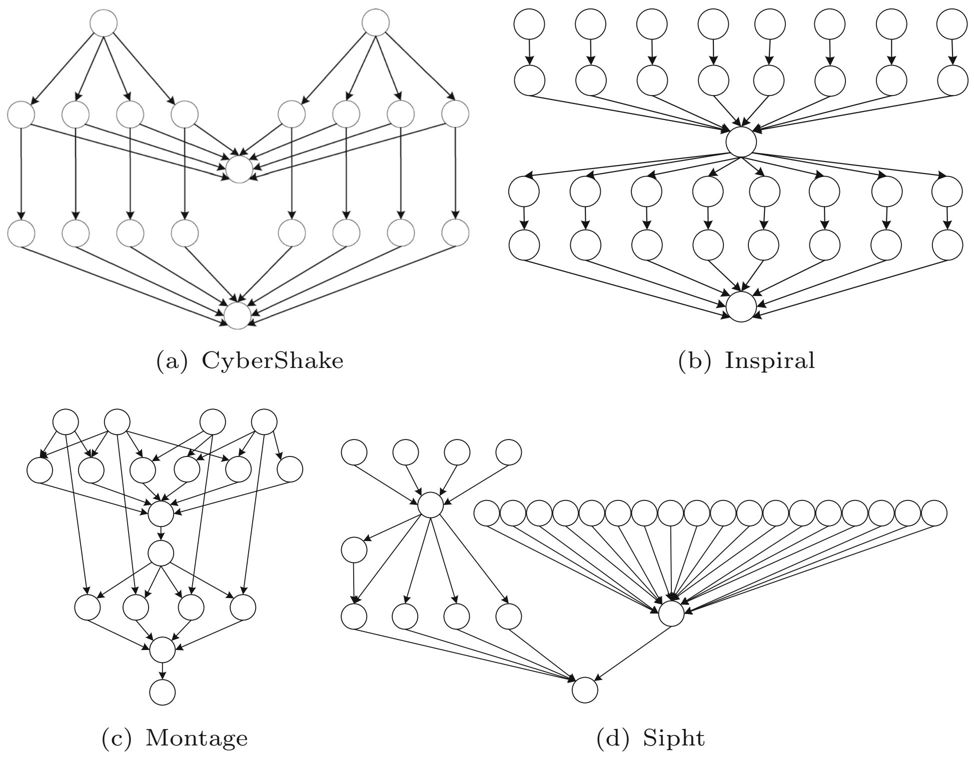


Fig. 4 The structures of the four workflows

make a fair comparison, all algorithms are coded by Java 1.8 and run on the server with two Intel(R) Xeon(R) E5-2650-V3@2.30GHz processors and 128 G of RAM under Linux CentOS release 6.5.

The experiment is made up of three aspects: (1) Parameter calibration of KMOEDA; (2) Ablation analysis of the components; (3) Comparison of KMOEDA and related multi-objective algorithms. Since RCMOWSP was not considered in previous research, the reliability constraint is established as follows.

$$\delta_{Rel} = \prod_{i=1}^{n} \max(\{rel(t_i, p_k) | p_k \in I\}) \times 75\% \tag{25}$$

where $rel(t_i, p_k)$ represents the reliability of task t_i on the virtual machines with the type of p_k. We set the reliability constraint as the 75% of maximum reliability. Moreover, because KMOEDA and the compared algorithms are stochastic approaches, all algorithms run independently for 30 times. Considering that the scale of problem space increases with the increase of workflow size, the termination criterion of each run is set to $100n$.

5.2 Performance metrics

Due to the considered RCMOWSP is essentially a multi-objective optimization problem, It is difficult to evaluate the performance of algorithms based on a comparison of isolate schedules. hypervolume and set coverage [48] are two popular performance metrics for evaluating the multi-objective optimization algorithms. Therefore, this paper applies hypervolume and set coverage to measure the non-dominated solutions obtained by the implemented algorithms.

(1) Hypervolume. It measures quality of a given Pareto set S by calculating the hypervolume of points. A reference point (1.2, 1.2) is employed to close the volume. Therefore, the closer the hypervolume value is to 1.44, the better S is. Then, *TET* and *TEC* are normalized to a number within [0, 1]. Next, the hypervolume is calculated as follows.

$$hypervolume(S) = \sum_{i=1}^{2} \sum_{j=1}^{|S|} \frac{f_i(s_j) - f_i^{min}}{f_i^{max} - f_i^{min}} \tag{26}$$

where s_j is the jth solution in S. $f_1(s_j)$ and $f_2(s_j)$ are *TET* and *TEC* of the solution s_j. f_i^{min} and f_i^{max} are the minimum and maximum values of ith objective.

(2) Set Coverage (C (A, B)). It reflects the dominance relationship between the two Pareto set A and B. The set coverage is calculated as follows.

$$C(A,B) = \frac{|\{b \in B | \exists a \in A, a \preceq b\}|}{|B|} \tag{27}$$

where A and B are two Pareto sets. The closer C (A, B) is to 1, the better the quality of A.

5.3 Parameter calibration

The parameters of the KMOEDA are calibrated using the full factorial design of the experiment. The levels of the parameters of KMOEDA are the population size $ps = \{5, 10, 15, 20, 25, 30\}$, $rs = \{10, 20, 30, 40, 50, 60\}$, $d = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$, and $\alpha = \{0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. It would result in $6 \times 6 \times 6 \times 6 = 1296$ configurations. All configurations are run on the Montage workflow with 25 tasks, and each configuration run 5 times independently. Table 3 provides the analysis of variance (ANOVA) results of hypervolume. The p-values < 0.05 are marked bold to demonsrate that the corresponding parameter is significant to the KMOEDA.

As shown in Table 3, ps, rs, and d have a significant effect on the performance of KMOEDA. The size of the external archive rs has the largest impact on the performance of KMOEDA because it has the highest F-Ratio. From the main effect plot of rs, it can be seen that the best value of rs is 30, and the worst value of rs is 10. The small

Table 3 ANOVA Results for parameters calibration

Parameters	Sum of squares	D.f	Mean square	F-ratio	p-value
ps	0.00059	5	0.00012	6.71	**0.0000**
rs	0.14232	5	0.02846	1633.21	**0.0000**
α	0.00011	5	0.00002	1.28	0.2692
d	0.00532	5	0.00106	61.04	**0.0000**
$ps * rs$	0.0005	25	0.00002	1.14	0.2849
$ps * \alpha$	0.00039	25	0.00002	0.9	0.6018
$ps * d$	0.00109	25	0.00004	2.51	**0.0001**
$rs * \alpha$	0.00024	25	0.00001	0.56	0.9598
$rs * d$	0.00851	25	0.00034	19.54	**0.0000**
$\alpha * d$	0.00059	25	0.00002	1.36	0.1124

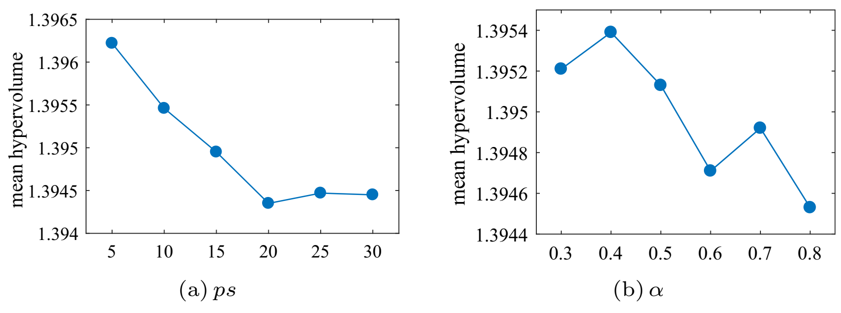
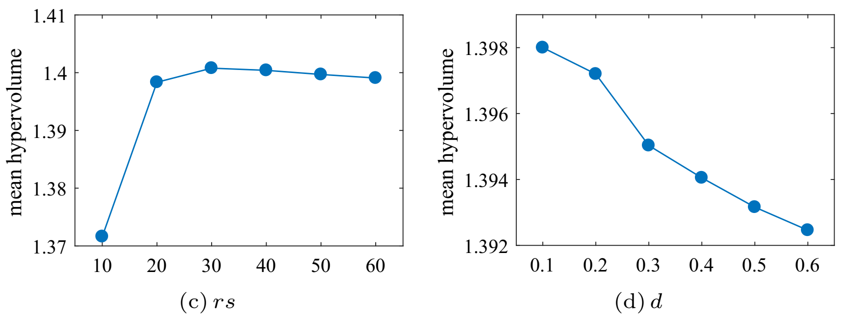



Fig. 5 Main effect plots of parameters

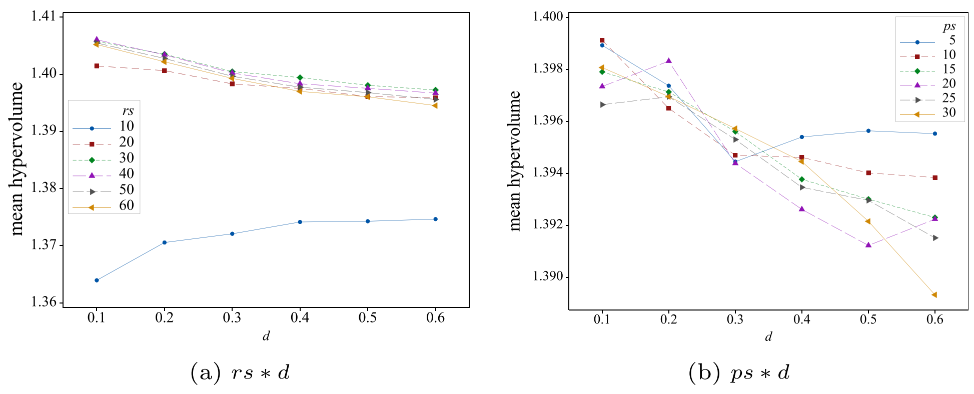


Fig. 6 Interaction effect plots between parameters

Table 4 Average hypervolume of ablation analysis

Workflows	KNI	KNG	KNR	KNL	KMOEDA
CyberShake_100	1.2896	1.0089	1.3999	1.4008	**1.4115**
Inspiral_100	1.0604	1.0583	1.2770	1.2607	**1.3020**
Montage_100	1.3225	0.6161	1.4141	1.4168	**1.4172**
Sipht_100	1.2471	1.3913	1.4216	1.4060	**1.4257**

external archive results in insufficient distribution of the non-dominated solution set, and then the probability model cannot reflect the objective space. Additionally, the small external archive will lead to insufficient reference templates, which will significantly reduce the diversity of the population. The over large external archive can accommodate more non-dominated solutions with the reduced convergence speed. As seen from Table 3, the interactions between *rs* and *d* have a significantly impact on the performance of KMOEDA because the corresponding *p*-value is smaller than 0.05. Therefore, the value of *rs* is set to 40 according to the interaction effect plot.

From Table 3, the *p*-value of *d* is also smaller than 0.05, and its F-Ratio is only smaller than that of *rs*. Therefore, *d* has a significant effect on the performance of KMOEDA. To be specific, the over large value of *d* means that more tasks should be re-deployed according to the probability model, and then reduce the convergence speed. Therefore, it can be seen from Fig. 5 that the best value of *d* is 0.1. From Table 3, the interaction relationships between *d* and *rs*, *d* and *ps* have significant impact on the performance of KMOEDA. Since the interaction between *rs* and *d* has a larger F-Ratio, the value of *d* is set to 0.1 according to the interaction effect plot.

From Table 3, the impact of *ps* on the performance of KMOEDA ranks third. The large size of the population will result in fewer probability model updates, which will lower the accuracy of the probability model. As depicted in Fig. 5, KMOEDA performs best with a small *ps* of 5.

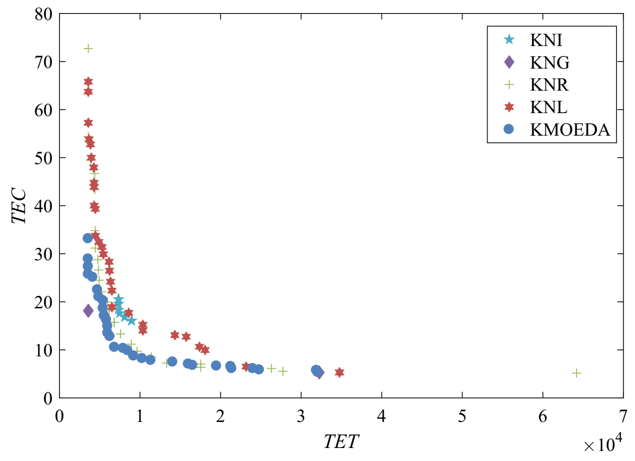


Fig. 7 Pareto fronts of KMOEDA and its variants on Sipht with 100 tasks

Additionally, the value of *ps* is determined by the interaction relationship between *ps* and *d* because it significantly affects the performance of KMOEDA. As seen from Fig. 6, the best value of *ps* is 10. Moreover, the learning rate α has no significant effect on the performance of KMOEDA, and it has no interaction effect with other parameters. Therefore, the value of α is set to 0.4 according to Fig. 5. As above results and analysis, the parameters are set as follows: $ps = 10$, $rs = 40$, $d = 0.1$, and $\alpha = 0.4$.

5.4 Ablation analysis

The impact of the core components on the performance of KMOEDA is further compared and examined in this section. The core components of KMOEDA include the initialization strategy, global search strategy, reliability-aware local search strategy, and elite enhancement strategy. Therefore, four variants of KMOEDA are constructed based on the above four components to be compared with KMOEDA. To verify the effect of the initialization strategy, a variant KNI replaces the initialization strategy with a random initialization approach. KNG removes the global search strategy to evaluate the effect of the global search strategy on performance. KNL removes the elite enhancement strategy to test the effect of the elite enhancement strategy on the performance of KMOEDA. KNR removes the reliability-aware local search strategy to confirm the effectiveness of the local search strategy. The above variants and KMOEDA are run 30 times independently with the same experimental environment and termination criterion. The CyberShake, Inspiral, Montage, and Sipht with 100 tasks are carried out to be the benchmark. The experimental results are reported in Table 4. The maximum values of hypervolume in each row are marked bold.

It can be seen from Table 4 that the average hypervolume of KMOEDA is superior to the other variants. Therefore, all of the components contribute to the performance of KMOEDA. To be specific, KNG performs the poorest, which demonstrates that the global search strategy significantly enhances the performance of KMOEDA. The performance of KNI is marginally better than KNG but poorer than KNR and KNL. The reason is that the distribution of the initialization solutions within the problem space has a significant effect on the probability model, and then the poor initialization solutions will reduce the effectiveness of the global search strategy. The performance of KNR and KNL are similar, but both are inferior to KMOEDA. This demonstrates the reliability-aware local search strategy and the elite enhancement strategy can effectively enhance the performance of KMOEDA.

As the above experimental results and analysis, KMOEDA can balance diversity and convergence well by using the components. The Pareto front obtained by KMOEDA and other variants on the Sipht workflow with 100 tasks is depicted in Fig. 7.

5.5 Comparison of other related algorithms

This section compares the proposed KMOEDA with five multi-objective optimization algorithms: EMS-C [16], NSGA-II [46], MOHEFT [35] ch-PICEA-g [38], and I_MaOPSO [39] to verify the effectiveness of KMOEDA on RCMOWSP. MOHEFT is a heuristic for solving the bi-objective workflow scheduling problem in the cloud environment. EMS-C, ch-PICEA-g, and I_MaOPSO are meta-heuristics for solving the multi-objective workflow scheduling problem in the cloud environment. NSGA-II is a popular multi-objective optimization algorithm. Therefore, the effectiveness of KMOEDA for solving the RCMOWSP can be adequately verified when compared to the above algorithms.

The benchmark is made up of below workflows: (1) CyberShake workflow with 30, 50, 100, and 1,000 tasks; (2) Inspiral workflow with 30, 50, 100, and 1,000 tasks; (3) Montage workflow with 25, 50, 100, 1,000 tasks; (4) Sipht workflow with 30, 60, 100, and 1,000 tasks. The aforementioned workflows are frequently carried out to test the performance of workflow scheduling algorithms. To make a fair comparison, all algorithms share the same termination criterion, which is the maximum number of function evaluations. Additionally, all algorithms were run 30 times independently. The average hypervolume and set coverage are displayed in Tables 5, 6, 7. The maximum values of hypervolume and set coverage are marked bold. In Tables 6 and 7, $C(-,A)$ represents the coverage of KMOEDA to the compared algorithm A, and $C(A,-)$ denotes the coverage of the compared algorithm A to

Table 5 Average hypervolume of KMOEDA and compared algorithms

Workflows	EMS-C	NSGA-II	MOHEFT	ch-PICEA-g	I_MaOPSO	KMOEDA
CyberShake_30	1.4025	1.3466	1.3902	1.3476	1.4015	**1.4238**
CyberShake_50	1.3823	1.3668	1.3248	1.3589	1.3704	**1.4143**
CyberShake_100	1.3945	1.3868	1.307	1.2482	1.3727	**1.4115**
CyberShake_1000	1.2874	0.3845	-	0.2699	1.2688	**1.3595**
Inspiral_30	1.3201	0.4245	1.2355	0.9503	1.2842	**1.3799**
Inspiral_50	1.2704	0.2526	1.2918	0.8619	1.231	**1.3488**
Inspiral_100	1.1926	0.2601	1.1642	0.5092	1.1918	**1.3189**
Inspiral_1000	0.6981	0.2419	-	0.2471	0.6305	**1.0252**
Montage_25	1.391	1.2857	1.1792	1.2481	1.3737	**1.4057**
Montage_50	1.4092	1.3131	1.2898	1.302	1.3927	**1.4168**
Montage_100	**1.4215**	1.3429	1.2729	1.3908	1.4087	1.4172
Montage_1000	1.3741	0.7211	-	0.9595	**1.3876**	1.3847
Sipht_30	1.4216	1.3643	1.4219	1.3356	1.4124	**1.4286**
Sipht_60	1.4185	1.3849	1.4237	1.3733	1.393	**1.4322**
Sipht_100	1.3782	1.3965	1.408	1.2882	1.3461	**1.4257**
Sipht_1000	1.2351	1.2974	-	1.1937	0.9485	**1.3164**

Table 6 Average set coverage between KMOEDA and compared algorithms

Workflows	C(-,EMS-C)	C(-,NSGA-II)	C(-,MOHEFT)	C(-,ch-PICEA-g)	C(-,I_MaOPSO)
CyberShake_30	**90.96%**	**86.73%**	**58.67%**	**98.72%**	**95.69%**
CyberShake_50	**82.33%**	**71.34%**	**52.67%**	**95.40%**	**95.41%**
CyberShake_100	**60.48%**	**58.72%**	**40.00%**	**72.61%**	**93.53%**
CyberShake_1000	**24.07%**	**51.29%**	**-**	**77.02%**	**80.19%**
Inspiral_30	**99.19%**	**100.00%**	**98.00%**	**96.46%**	**90.86%**
Inspiral_50	**94.07%**	**100.00%**	**62.00%**	**91.61%**	**99.74%**
Inspiral_100	**76.44%**	**100.00%**	**61.33%**	**86.63%**	**80.14%**
Inspiral_1000	**42.57%**	**100.00%**	**-**	**100.00%**	**100.00%**
Montage_25	**83.40%**	**100.00%**	**72.00%**	**91.98%**	**93.70%**
Montage_50	**60.19%**	**97.91%**	**50.00%**	**48.20%**	**95.97%**
Montage_100	37.46%	**55.05%**	26.67%	6.73%	**80.08%**
Montage_1000	26.43%	**23.15%**	-	7.40%	**58.74%**
Sipht_30	**79.48%**	**78.91%**	22.67%	**100.00%**	**92.00%**
Sipht_60	**51.71%**	**78.51%**	**40.00%**	**99.92%**	**100.00%**
Sipht_100	**60.44%**	**49.85%**	**28.67%**	**100.00%**	**100.00%**
Sipht_1000	**47.93%**	**38.94%**	**-**	**39.27%**	**98.60%**

KMOEDA. Due to the high space complexity and memory requirement of MOHEFT, the experimental results of MOHEFT on the extra large-scale workflows are not provided.

It can be observed from Table 5 that the average hypervolume obtained by KMOEDA is superior to that of the other compared algorithms on most of the instances. However, the hypervolume of KMOEDA is slightly smaller than that of EMS-C on the Montage workflow with 100 tasks. On the Montage with 1000 tasks, the hypervolume of KMOEDA is marginally lower than that of I_MaOPSO. Table 5 further shows that the gap between the hypervolume of KMOEDA and that of other algorithms rapidly widens with the size of workflow increases. The influence of reliability constraints on the problem space increases as the size of workflow increases. The problem-specific strategies of KMOEDA can effectively enhance performance of KMOEDA. Table 5 further demonstrates that the hypervolume of KMOEDA is higher than that of compared algorithms EMS-C, NSGA-II, MOHEFT, ch-PICEA-g, and I_MaOPSO by 4.34, 38.93, 6.65, 29.76, and 7.33%, respectively. The mean values of hypervolume with 95% LSD interval are shown in Fig. 8.

The average set coverage between KMOEDA and other compared algorithms is reported in Tables 6 and 7. From Tables 6 and 7, it can be observed that the KMOEDA has

Table 7 Average set coverage between compared algorithms and KMOEDA

Workflows	C(-,EMS-C)	C(-,NSGA-II)	C(-,MOHEFT)	C(-,ch-PICEA-g)	C(-,I_MaOPSO)
CyberShake_30	12.35%	3.73%	12.17%	0.12%	11.73%
CyberShake_50	14.35%	7.17%	8.33%	1.52%	14.11%
CyberShake_100	18.77%	12.17%	37.89%	3.03%	16.45%
CyberShake_1000	20.37%	1.40%	-	0.00%	11.15%
Inspiral_30	5.16%	0.24%	0.47%	0.62%	5.30%
Inspiral_50	3.59%	0.23%	27.87%	1.66%	2.74%
Inspiral_100	8.41%	0.00%	14.61%	0.94%	6.44%
Inspiral_1000	7.12%	0.00%	-	0.00%	2.80%
Montage_25	18.30%	3.63%	23.60%	1.35%	17.44%
Montage_50	23.21%	5.37%	22.73%	13.90%	19.25%
Montage_100	**58.80%**	19.43%	**41.94%**	**39.14%**	28.71%
Montage_1000	**40.24%**	16.07%	**-**	**17.86%**	43.45%
Sipht_30	17.82%	7.87%	**52.06%**	0.00%	3.71%
Sipht_60	31.02%	9.28%	27.34%	0.12%	1.36%
Sipht_100	26.05%	23.48%	26.30%	0.00%	0.50%
Sipht_1000	31.40%	38.80%	-	0.00%	0.00%

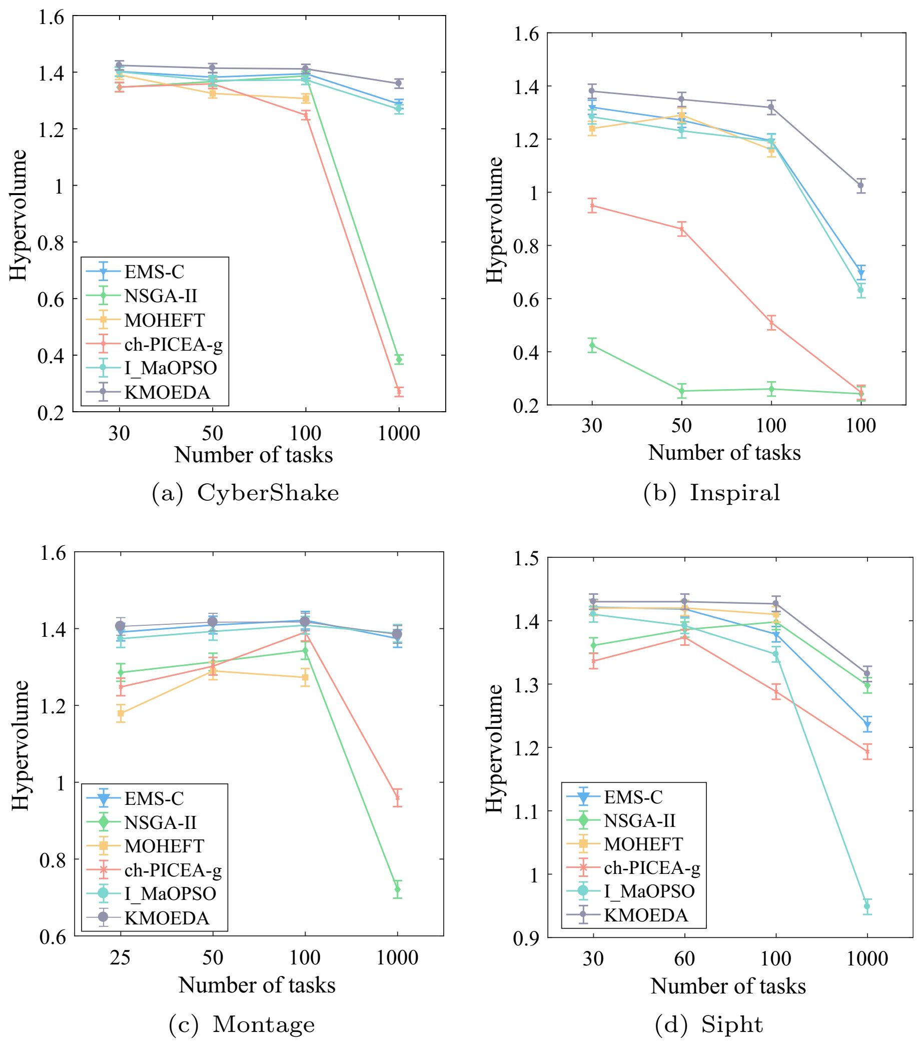


Fig. 8 Mean value of hypervolume with 95% LSD interval

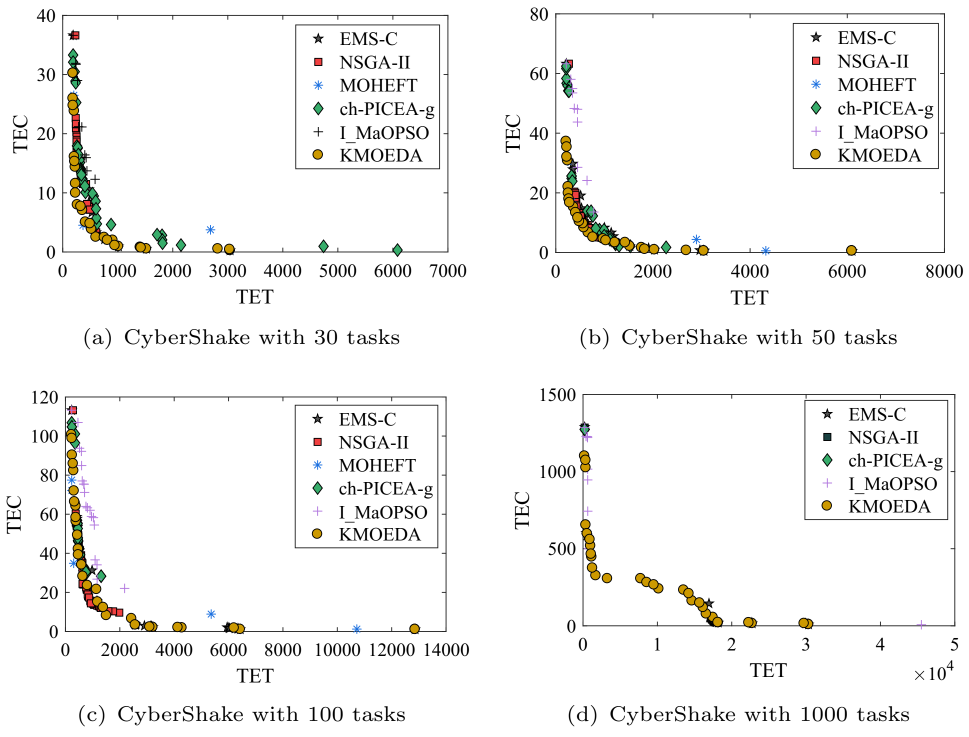


Fig. 9 Pareto fronts of KMOEDA and other algorithms on CyberShake

Table 8 Statistical results of Wilcoxon's test

KMOEDA vs	R+	R-	Z	*p*-value
EMS-C	135	1	−3.464488	**0.000531**
NSGA-II	136	0	−3.516196	**0.000438**
MOHEFT	78	0	−3.059412	**0.002218**
ch-PICEA-g	153	0	−3.5162	**0.000438**
I_MaOPSO	135	1	−3.46449	**0.000531**

more set coverage on other compared algorithms than the compared algorithms do non most test instances. On the Montage with 100 and 1,000 tasks, the set coverage of KMOEDA is lower than less than EMS-C and chi-PIECA-g, respectively. On the Montage with 100 tasks and the Sipht with 30 tasks instances, the set coverage of KMOEDA is smaller than MOHEFT. However, the experimental results in most of the instances still demonstrate the superior performance of KMOEDA. The Pareto fronts obtained by KMOEDA and other compared algorithms are shown in Fig. 9.

To further investigate the performance disparity between KMOEDA and the compared algorithms, we apply Wilcoxon's test to verify the significant difference between the comparison algorithms. The results of Wilcoxon's test are displayed in Table 8. The *p*-values < 0.05 are marked bold to demonstrate that the KMODEA is significantly superior to the compared algorithms. Since the *p*-values between KMOEDA and other compared algorithms are less than 0.05, it can be concluded that KMOEDA is significantly superior to EMS-C, NSGA-II, MOHEFT, ch-PICEA-g, and I_MaOPSO with 95% confidence level.

Based on the experimental results and analysis, it can be observed that the proposed KMOEDA can effectively solve the RCMOWSP. In fact, the proposed KMOEDA still follows the current universal framework of meta-heuristics, which includes initialization strategy, global search strategy, and local search strategy. However, in KMOEDA, the global and local strategies are design based on the characteristics of the problem, especially for reliability-aware local search strategies. The collaboration of these problem-specific strategies can effectively contribute to improving the convergence and diversity of non-dominated solutions when compared to the other existing algorithms for the related problems.

6 Conclusions and future work

This paper proposes a knowledge-based multi-objective estimation of distribution algorithm (KMOEDA) to solve a novel workflow scheduling problem, i.e., reliability-constrained multi-objective workflow scheduling problem (RCMOWSP) with the objectives of minimizing execution time and cost subject to reliability constraint. The computational experiments and comparative results with several related algorithms demonstrate that KMOEDA is more effective in solving RCMOWSP whether on small-scale or large-scale workflows. The superior performance of

KMOEDA is attributed to the aspects below: 1) The initialization strategy constructs the initial solutions in a promising area. 2) The global search strategy uses the probability model to maintain the diversity of the population well. 3) The reliability-aware search strategy can effectively handle unfeasible solutions. 4) The elite enhancement strategy performs well based on the elite solutions in the external archive.

The following three directions will be investigated in subsequent studies. First, the multi-objective workflow scheduling problem with multi-billing mechanism will be deeply studied. Second, the real-time fault-tolerant large-scale workflow scheduling problem should be further researched. Additionally, it is necessary to apply machine learning approaches to solve the multi-objective workflow scheduling problem in a cloud environment.

Author contributions ML and SQ wrote the main manuscript text and prepared all figures. All authors reviewed the manuscript.

Funding Not applicable.

Data availability The data presented in this study are available on request from the corresponding author.

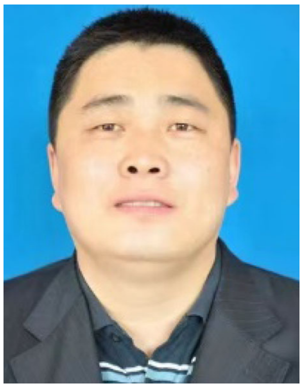
Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Basu, A., Kumar, A.: Research commentary: workflow management issues in e-business. Info. Syst. Res. **13**(1), 1–14 (2002)
2. Berriman, G.B., Deelman, E., Good, J.C., Jacob, J.C., Katz, D.S., Kesselman, C., Laity, A.C., Prince, T.A., Singh, G., Su, M.-H.: Montage: a grid-enabled engine for delivering custom science-grade mosaics on demand. In: Optimizing Scientific Return for Astronomy Through Information Technologies, pp. 221–232. SPIE (2004)
3. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: an overview of workflow system features and capabilities. Future Gener. Comput. Syst. **25**(5), 528–540 (2009)
4. Rizvi, N., Ramesh, D.: Hbdcws: heuristic-based budget and deadline constrained workflow scheduling approach for heterogeneous clouds. Soft Comput. **24**(24), 18971–18990 (2020)
5. Du, M., Wang, Y., Ye, K., Xu, C.: Algorithmics of cost-driven computation offloading in the edge-cloud environment. IEEE Trans. Comput. **69**(10), 1519–1532 (2020)
6. Lin, B., Huang, Y., Zhang, J., Hu, J., Chen, X., Li, J.: Cost-driven off-loading for DNN-based applications over cloud, edge, and end devices. IEEE Trans. Ind. Info. **16**(8), 5456–5466 (2019)
7. Ye, K., Shen, H., Wang, Y., Xu, C.: Multi-tier workload consolidations in the cloud: profiling, modeling and optimization. IEEE Trans. Cloud Comput. (2020)
8. Lin, B., Zhu, F., Zhang, J., Chen, J., Chen, X., Xiong, N.N., Mauri, J.L.: A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing. IEEE Trans. Ind. Info. **15**(7), 4254–4265 (2019)
9. Wang, W., Jiang, Y., Wu, W.: Multiagent-based resource allocation for energy minimization in cloud computing systems. IEEE Trans. Syst. Man Cybern. Syst. **47**(2), 205–220 (2016)
10. Sousa, E., Lins, F., Tavares, E., Cunha, P., Maciel, P.: A modeling approach for cloud infrastructure planning considering dependability and cost requirements. IEEE Trans. Syst. Man Cybern. Syst. **45**(4), 549–558 (2014)
11. Shirvani, M.H.: A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. Eng. Appl. Artif. Intel. **90**, 103501 (2020)
12. Qin, S., Pi, D., Shao, Z., Xu, Y.: A knowledge-based adaptive discrete water wave optimization for solving cloud workflow scheduling. IEEE Trans. Cloud Comput. (2021)
13. Qin, S., Pi, D., Shao, Z., Xu, Y.: Hybrid collaborative multi-objective fruit fly optimization algorithm for scheduling workflow in cloud environment. Swarm Evol. Comput. **68**, 101008 (2022)
14. Zhu, J., Li, X., Ruiz, R., Li, W., Huang, H., Zomaya, A.Y.: Scheduling periodical multi-stage jobs with fuzziness to elastic cloud resources. IEEE Trans. Parallel Distrib. Syst. **31**(12), 2819–2833 (2020)
15. Li, Z., Chang, V., Hu, H., Hu, H., Li, C., Ge, J.: Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds. Info. Sci. **568**, 13–39 (2021)
16. Zhu, Z., Zhang, G., Li, M., Liu, X.: Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1344–1357 (2015)
17. Faragardi, H.R., Sedghpour, M.R.S., Fazliahmadi, S., Fahringer, T., Rasouli, N.: Grp-heft: a budget-constrained resource provisioning scheme for workflow scheduling in IAAS clouds. IEEE Trans. Parallel Distrib. Syst. **31**(6), 1239–1254 (2019)
18. Jia, Y.-H., Chen, W.-N., Yuan, H., Gu, T., Zhang, H., Gao, Y., Zhang, J.: An intelligent cloud workflow scheduling system with time estimation and adaptive ant colony optimization. IEEE Trans. Syst. Man Cybern. Syst. **51**(1), 634–649 (2018)
19. Li, H., Wang, D., Zhou, M., Fan, Y., Xia, Y.: Multi-swarm co-evolution based hybrid intelligent optimization for bi-objective multi-workflow scheduling in the cloud. IEEE Trans. Parallel Distrib. Syst. **33**(9), 2183–2197 (2021)
20. Wang, Y., Zuo, X.: An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules. IEEE/CAA J. Automatica Sinica **8**(5), 1079–1094 (2021)
21. Garg, R., Mittal, M., Son, L.H.: Reliability and energy efficient workflow scheduling in cloud environment. Cluster Comput. **22**(4), 1283–1297 (2019)
22. Topcuoglu, H., Hariri, S., Wu, M.-Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
23. Wang, Z., Hayat, M.M., Ghani, N., Shaban, K.B.: Optimizing cloud-service performance: efficient resource provisioning via optimal workload allocation. IEEE Trans. Parallel Distrib. Syst. **28**(6), 1689–1702 (2016)
24. Topcuoglu, H., Hariri, S., Wu, M.-Y.: Task scheduling algorithms for heterogeneous processors. In: Proceedings Eighth Heterogeneous Computing Workshop (HCW'99), pp. 3–14. IEEE (1999)
25. De Coninck, E., Verbelen, T., Vankeirsbilck, B., Bohez, S., Simoens, P., Dhoedt, B.: Dynamic auto-scaling and scheduling of deadline constrained service workloads on IAAS clouds. J. Syst. Softw. **118**, 101–114 (2016)
26. Duan, R., Prodan, R., Li, X.: Multi-objective game theoretic scheduling of bag-of-tasks workflows on hybrid clouds. IEEE Trans. Cloud Comput. **2**(1), 29–42 (2014)
27. Li, S., Zhou, Y., Jiao, L., Yan, X., Wang, X., Lyu, M.R.-T.: Towards operational cost minimization in hybrid clouds for dynamic resource provisioning with delay-aware optimization. IEEE Trans. Serv. Comput. **8**(3), 398–409 (2015)

28. Bittencourt, L.F., Madeira, E.R., Da Fonseca, N.L.: Scheduling in hybrid clouds. IEEE Commun. Magaz. **50**(9), 42–47 (2012)
29. Meng, S., Huang, W., Yin, X., Khosravi, M.R., Li, Q., Wan, S., Qi, L.: Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications. IEEE Trans. Ind. Info. **17**(6), 4219–4228 (2020)
30. Lu, P., Sun, Q., Wu, K., Zhu, Z.: Distributed online hybrid cloud management for profit-driven multimedia cloud computing. IEEE Trans. Multimedia **17**(8), 1297–1308 (2015)
31. Zhu, J., Li, X., Ruiz, R., Xu, X.: Scheduling stochastic multi-stage jobs to elastic hybrid cloud resources. IEEE Trans. Parallel Distrib. Syst. **29**(6), 1401–1415 (2018)
32. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
33. Wang, Z.-J., Zhan, Z.-H., Yu, W.-J., Lin, Y., Zhang, J., Gu, T.-L., Zhang, J.: Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. IEEE Trans. Cybern. **50**(6), 2715–2729 (2019)
34. Wu, Q., Ishikawa, F., Zhu, Q., Xia, Y., Wen, J.: Deadline-constrained cost optimization approaches for workflow scheduling in clouds. IEEE Tran. Parallel Distrib. Syst. **28**(12), 3401–3412 (2017)
35. Durillo, J.J., Prodan, R.: Multi-objective workflow scheduling in amazon EC2. Cluster Comput. **17**(2), 169–189 (2014)
36. Wu, Q., Zhou, M., Zhu, Q., Xia, Y., Wen, J.: Moels: Multiobjective evolutionary list scheduling for cloud workflows. IEEE Trans. Autom. Sci. Eng. **17**(1), 166–176 (2019)
37. Choudhary, A., Gupta, I., Singh, V., Jana, P.K.: A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Future Gener. Comput. Syst. **83**, 14–26 (2018)
38. Paknejad, P., Khorsand, R., Ramezanpour, M.: Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment. Future Gener. Comput. Syst. **117**, 12–28 (2021)
39. Saeedi, S., Khorsand, R., Bidgoli, S.G., Ramezanpour, M.: Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing. Comput. Ind. Eng. **147**, 106649 (2020)
40. Chen, Z.-G., Zhan, Z.-H., Lin, Y., Gong, Y.-J., Gu, T.-L., Zhao, F., Yuan, H.-Q., Chen, X., Li, Q., Zhang, J.: Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach. IEEE Trans. Cybern. **49**(8), 2912–2926 (2018)
41. Larrañaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, vol. 2. Springer (2001)
42. Baluja, S.: Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Technical report, Carnegie-Mellon Universiy Pittsburgh, PA, Department of Computer Science (1994)
43. Shao, W., Pi, D., Shao, Z.: A pareto-based estimation of distribution algorithm for solving multi-objective distributed no-wait flow-shop scheduling problem with sequence-dependent setup time. IEEE Trans. Autom. Sci. Eng. **16**(3), 1344–1360 (2019)
44. Shao, W., Shao, Z., Pi, D.: Modeling and multi-neighborhood iterated greedy algorithm for distributed hybrid flow shop scheduling problem. Knowl.-Based Syst. **194**, 105527 (2020)
45. Zhao, F., He, X., Wang, L.: A two-stage cooperative evolutionary algorithm with problem-specific knowledge for energy-efficient scheduling of no-wait flow-shop problem. IEEE Trans. Cybern. **51**(11), 5291–5303 (2020)
46. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)
47. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. **29**(3), 682–692 (2013)
48. Zitzler, E., Thiele, L.: Multi-objective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evol. Comput. **3**(4), 257–271 (1999)



Ming Li is currently pursuing Ph.D. degree in computer science and technology from Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research interests include deep learning, machine learning, and computer vision.

Dechang Pi received the B.Eng. and M.Eng. degree from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1994 and 1997, respectively, his Ph.D. degree in computer science and technology from Nanjing University of Aeronautics and Astronautics, in 2002. Now he is a professor and Ph.D. supervisor in Nanjing University of Aeronautics and Astronautics. His research interests are data mining, big data analysis in manufacturing, and intelligent optimization methods.

Shuo Qin received the B.Eng. degree and the M.Eng. degree in software engineering from LanZhou University of Technology , Lanzhou, in 2016 and 2019, respectively, he is currently pursuing Ph.D. degree in computer science and technology from Nanjing University of Aeronautics and Astronautics, Nanjing, China. His current research interests include cloud computing and intelligent optimization methods.