Complex & Intelligent Systems (2023) 9:7337–7348
https://doi.org/10.1007/s40747-023-01137-w

ORIGINAL ARTICLE

Decision variable contribution based adaptive mechanism for evolutionary multi-objective cloud workflow scheduling

Jun Li[1] · Lining Xing[2] · Wen Zhong[3] · Zhaoquan Cai[4,5] · Feng Hou[6]

Received: 6 December 2022 / Accepted: 8 May 2023 / Published online: 29 June 2023

Abstract

Workflow scheduling is vital to simultaneously minimize execution cost and makespan for cloud platforms since data dependencies among large-scale workflow tasks and cloud workflow scheduling problem involve large-scale interactive decision variables. So far, the cooperative coevolution approach poses competitive superiority in resolving large-scale problems by transforming the original problems into a series of small-scale subproblems. However, the static transformation mechanisms cannot separate interactive decision variables, whereas the random transformation mechanisms encounter low efficiency. To tackle these issues, this paper suggests a decision-variable-contribution-based adaptive evolutionary cloud workflow scheduling approach (VCAES for short). To be specific, the VCAES includes a new estimation method to quantify the contribution of each decision variable to the population advancement in terms of both convergence and diversity, and dynamically classifies the decision variables according to their contributions during the previous iterations. Moreover, the VCAES includes a mechanism to adaptively allocate evolution opportunities to each constructed group of decision variables. Thus, the decision variables with a strong impact on population advancement are assigned more evolution opportunities to accelerate population to approximate the Pareto-optimal fronts. To verify the effectiveness of the proposed VCAES, we carry out extensive numerical experiments on real-world workflows and cloud platforms to compare it with four representative algorithms. The numerical results demonstrate the superiority of the VCAES in resolving cloud workflow scheduling problems.

Keywords Cloud computing · Workflow scheduling · Multi-objective · Evolutionary optimization · Large-scale

Introduction

Workflows have been commonly used to describe data processing applications from diversified fields, such as the Internet of Things and bio-informatics [1–3]. These workflows often comprise of large-scale data-dependent tasks, which are computation and data intensive. Then, executing various workflow applications calls for powerful high-performance infrastructures. With substantial advantages, such as economies of scale, on demand supply of resources, high elasticity, and reliability, cloud computing is attracting more and more enterprises or individuals to deploy their big data processing workflows [4, 5].

Workflow scheduling in cloud computing is a key technology for achieving the reduction of both execution cost and makespan to gain more profits for cloud providers and ensure the quality of service for cloud consumers [6]. Workflow scheduling problem involves determining the mappings from tasks to resources and the task order on each resource, and is a classic NP-complete [7, 8]. Also, the execution cost

✉ Lining Xing
lnxing@xidian.edu.cn

✉ Wen Zhong
zhongwen_nudt@163.com

1 School of Management, Hunan Institute of Engineering, Xiangtan 411104, Hunan, People's Republic of China

2 School of Electronic Engineering, Xidian University, Xi'an 710071, Shaanxi, People's Republic of China

3 Department of Traffic Management, Hunan Police Academy, Changsha 410138, Hunan, People's Republic of China

4 Shanwei Institute of Technology, Shanwei 516600, Guangdong, People's Republic of China

5 School of Computer Science and Engineering, Huizhou University, Huizhou 516007, Guangdong, People's Republic of China

6 School of Mathematical and Computational Sciences, Massey University, Albany 4442, New Zealand

and makespan of workflow scheduling are two conflicting optimization objectives [9]. So far, multi-objective evolutionary algorithms have become popular to search a set of compromise solutions within an acceptable time [10–12]. To solve the workflow scheduling problem in cloud computing, some studies design new evolution and selection operators to improve the classical multi-objective evolutionary algorithms.

Over the past decade or so, designing efficient evolution operators to reproduce new solutions for multi-objective workflow scheduling problem has attracted considerable research interest [13]. First of all, the popular list-based workflow scheduling methods were embedded into the multi-objective evolutionary optimization framework as evolution operators [14–16]. Secondly, bio-inspired evolution operators, such as artificial neural network [17–19], ant colony optimization [20], firefly algorithm [21], particle swarm optimization [22–24], and grey wolf optimization [25], were modified as evolution operators to solve the multi-objective workflow scheduling problem. Thirdly, integrating heuristic rules and bio-inspired optimization techniques to reproduce offspring populations has become a popular technological path. For instance, Choudhary et al. [26] combined the gravitational search method and the list-based workflow scheduling method to solve bi-objective workflow scheduling problem in cloud computing. Hosseini et al. [27] merged the simulated annealing and a task duplication strategy to optimize the makespan and execution cost of workflows. Mohammadzadeh et al. [28] integrated the antlion and grasshopper optimization algorithms to balance throughput, makespan, cost, and energy consumption of executing workflows in cloud platforms. Zhang et al. [29] enhanced the list-based workflow scheduling method with a local search mechanism to balance the makespan and energy consumption of workflow execution.

At the same time, some studies went into designing selection operators to balance multiple conflicting objectives of workflow execution in cloud computing. For example, Zhou et al. [30] merged a fuzzy-dominance-based environmental selection and a list-based workflow scheduling method to minimize execution cost and makespan of workflows in cloud computing. Kumar et al. [31] integrated the entropy weight mechanism into a multi-criteria decision-making framework to balance makespan, execution cost, reliability, and energy consumption. Ye et al. [32] improved a knee point driven evolutionary method to balance makespan, reliability, execution cost, and the mean durations of all workflow tasks. Pham et al. [33] focused on the volatility of spot cloud resources and improved the multi-objective evolutionary algorithm to make a trade-off between makespan and execution cost for workflows in cloud computing.

In the evolutionary optimization community, a multi-objective optimization problem is generally considered large-scale if it has at least one hundred decision variables [34]. The multi-objective cloud workflow scheduling involves hundreds or even thousands of decision variables, and is a typical large-scale multi-objective optimization problem. However, the existing relevant studies evolve all decision variables as a whole and allocate evolution opportunities to each variable equally. This results in the low efficiency of these existing studies.

The recent research results in the evolutionary computation community demonstrate that cooperative coevolution [34, 35] has become a crucial and effective way to solve large-scale multi-objective optimization problems. In cooperative coevolution approaches, all the decision variables are classified into multiple groups, and decision variables in different groups are evolved in a round-robin manner [36, 37]. These static classification techniques work well when problems' decision variables are fully or partially separable. However, this is not the case for multi-objective cloud workflow scheduling with nonseparable decision variables caused by data dependencies among tasks.

Besides, multi-objective cloud workflow scheduling poses an imbalance feature among decision variables regarding their contributions to optimization objectives. For instance, delaying the completion of a workflow task on the critical paths [38] often successively delays the completion of many tasks, including its successor tasks and other tasks being executed after the delayed tasks and their successor tasks. Whereas slightly delaying other tasks on the non-critical paths may not cause this chain reaction. The imbalance feature means that we should equip different decision variables with different evolution opportunities. This motivates us to design a decision variable contribution based adaptive mechanism to dynamically adjust the variable grouping and allocate evolution opportunities during the evolution process. Our main contributions in this paper are as follows.

- We define the contribution of a decision variable as the fitness improvement of the solution generated by perturbing this decision variable. Then, we try to dynamically measure the contribution of each decision variable and classify them according to their contributions.
- We design an adaptive mechanism to dynamically allocate more evolution opportunities for the variable groups with more contributions to generate offspring solutions efficiently.
- In the context of fifteen real-world workflows and the Amazon Elastic Compute Cloud, we compare the proposal with four state-of-the-art multi-objective cloud workflow scheduling algorithms. The results demonstrate the competitive performance of the proposal in simultaneously optimizing execution cost and makespan.

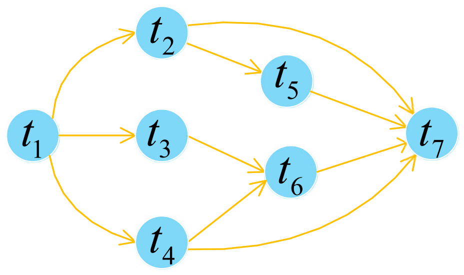

Fig. 1 An example workflow of seven tasks

This paper is organized as follows. In the second section formulates the multi-objective workflow scheduling problem. In the third section designs the proposed VCAES, followed by experimental verifications in the fourth section. In the final section concludes this paper.

Problem formulation

This section first describes the models of workflows and cloud resources, and then formulates the model for multi-objective workflow scheduling in cloud computing.

Workflow model

Without loss of generality, a workflow application is described by a Directed Acyclic Graph (DAG), whose vertices and directed edges represent workflow tasks and data dependencies, respectively. Formally, we construct the directed acyclic graph for a workflow application as $\Psi = \{T, D\}$, where $T = \{t_1, t_2, \ldots, t_n\}$ denotes the vertex set corresponding to task set, $D \subseteq T \times T$ denotes the edge set corresponding to data dependencies among tasks. The existence of an edge $d_{i,j} \in D$ means that t_j's start demands t_i's output results. Generally, task t_i is referred to as an direct predecessor of task t_j, and t_j is referred to as an direct successor of t_i. Regarding a task t_i, all its direct predecessors is expressed as a set $P(t_i)$, and all its direct successors is expressed as a set $S(t_i)$.

Figure 1 gives a visual example of a directed acyclic graph for a workflow with seven tasks, i.e., $T = \{t_1, t_2, \ldots, t_7\}$. An edge $d_{1,2}$ denotes the data dependency from t_1 to t_2, meaning that t_2's start have to wait for t_1's output results. In Fig. 1, regarding task t_6, the set of its direct predecessors is $P(t_6) = \{t_3, t_4\}$, and the set of its direct successors is $S(t_6) = \{t_7\}$.

Cloud resource model

This paper targets the popular cloud paradigm, i.e., Infrastructure as a Service (IaaS). In this paradigm, cloud providers offer multiple types of cloud resources on demand [39, 40]. The differences between different types of cloud resources mainly lie in their charging prices and performance configurations, such as number of CPU cores, memory size, and network bandwidth. Assuming that cloud platforms offer m types of resources, then we model them as $\Gamma = \{1, 2, \ldots, m\}$, where $\tau \in \Gamma$ denotes the τ-th resource type. Regarding a type τ, we employ $\mathrm{pr}(\tau)$ and $\mathrm{con}(\tau)$ to represent its price and configurations. Then, a cloud resource of type τ is modeled as $r_k^\tau = \{k, \mathrm{pr}(\tau), \mathrm{con}(\tau)\}$, where k denotes the index of resource r_k^τ.

Refer to well-known cloud providers (e.g., Amazon EC2[1] and Alibaba Cloud ECS[2]), this study follows the resource charging basis of pay-as-you-use. Under this rule, any consumer can rent any number of resources on demand and is charged according to the real usage time. In general, cloud resources are charged based on the number of billing periods, and the partial period will be rounded up to one more. In case that the period length is 60.0 min, the number of billing periods for 60.01 min is two.

Multi-objective scheduling cloud workflows

Since cloud resources are available on demand, we build a resource pool based on the maximum resource requirements for running a workflow. Assuming the maximum parallelism of the workflow is p, the resource pool includes p resources of each type. Then, we describe the resource pool as: $R = \left\{r_1^1, r_2^1, \ldots, r_p^1, r_{p+1}^2, r_{p+1}^2, \ldots, r_{2\cdot p}^2, \ldots, r_{m\cdot p}^m\right\}$.

The decision vector $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ is used to represent the mappings from workflow tasks to cloud resources, where the value of decision variable x_i is decoded as the index of the cloud resource mapped to the i-th task. It is worth noting that the value range of each decision variable is an integer from 1 to $m \cdot p$.

Given a decision vector, assume that the task t_i is mapped to resource r_k^τ. The start time $\mathrm{st}_{i,k}$ of task t_i refers to the maximum time of receiving all the input data and the available time of the mapping resource.

On resource r_k^τ, we assume the set of tasks being executed before task t_i as follows:

$$B_i = \left\{t_p | I(t_p) < I(t_i)\right\}, \tag{1}$$

where $I(t_p)$ indicates t_p's order number on resource r_k^τ.

Then, the start time $\mathrm{st}_{i,k}$ of task t_i on cloud resource r_k^τ can be described as follows:

$$\mathrm{st}_{i,k} = \max\left\{\max_{t_b \in B_i} \mathrm{ft}_{b,k}, \max_{t_p \in P(t_i)} \left\{\mathrm{ft}_{p,*} + \mathrm{dt}_{p,i}\right\}\right\}, \tag{2}$$

[1] https://aws.amazon.com/ec2/?nc2=h_ql_prod_fs_ec2.

[2] https://www.aliyun.com/.

where $\mathrm{f}t_{b,k}$ indicates t_b's finish time on resource r_k^τ, $\mathrm{f}t_{p,*}$ indicates the finish time of task t_p, and $\mathrm{d}t_{p,i}$ indicates the data transfer time from t_p to t_i.

Before scheduling, task t_i's execution time $\mathrm{e}t_{i,k}$ on cloud resource r_k^τ can be predicted by the computation length of task t_i and the CPU frequency of the mapped resource r_k^τ. The relationship among $\mathrm{s}t_{i,k}$, $\mathrm{e}t_{i,k}$, and $\mathrm{f}t_{i,k}$ is described as follows:

$$\mathrm{f}t_{i,k} = \mathrm{s}t_{i,k} + \mathrm{e}t_{i,k}. \tag{3}$$

Given a decision vector, the set of all tasks mapped to cloud resource r_k^τ can be formulated as:

$$T_k = \{t_i | x_i = k, i \in \{1, 2, \ldots, n\}\}. \tag{4}$$

With the task set T_k, the start time $\mathrm{u}t_k$ and end time $\mathrm{n}t_k$ of renting resource r_k^τ can be computed as follows:

$$\begin{aligned} \mathrm{u}t_k &= \min_{t_i \in T_k} \left\{ \mathrm{s}t_{i,k} - \max_{t_p \in P(t_i)} \mathrm{d}t_{p,i} \right\}, \\ \mathrm{n}t_k &= \max_{t_i \in T_k} \left\{ \mathrm{f}t_{i,k} + \max_{t_s \in S(t_i)} \mathrm{d}t_{i,s} \right\}. \end{aligned} \tag{5}$$

Based on the above analysis, the first optimization objective, i.e., minimizing the execution cost, can be formulated as follows:

$$\text{Min } f_1(\boldsymbol{x}) = \sum_{r_k^\tau \in R} \mathrm{pr}(\tau) \times \left\lceil \frac{\mathrm{n}t_k - \mathrm{u}t_k}{C} \right\rceil, \tag{6}$$

where C indicates the length of a billing period for cloud resources.

The second optimization objective of this paper is to minimize the workflow's makespan, which corresponds to the maximum finish time of all the tasks in this workflow. The second optimization objective can be formulated as follows:

$$\text{Min } f_2(\boldsymbol{x}) = \max_{t_i \in T} \mathrm{f}t_{i,*}. \tag{7}$$

Thus, the model for multi-objective workflow scheduling problem in cloud computing can be summarised as follows:

$$\begin{cases} \text{Min} \quad \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})], \\ \text{S.t.} \\ \qquad \boldsymbol{x} \in \{1, 2, \ldots, m \cdot p\}^n. \end{cases} \tag{8}$$

Pareto-dominance has been widely employed to compare solutions in the multi-objective optimization field.

Pareto-dominance: Assuming $\mathbf{x}_1$ and $\mathbf{x}_2$ are two feasible solutions. $\mathbf{x}_1$ is regarded to dominate $\mathbf{x}_2$ (denoted as $\mathbf{x}_1 \prec \mathbf{x}_2$) if and only if the two objectives of $\mathbf{x}_1$ are not inferior to that of $\mathbf{x}_2$ (i.e., $f_j(\mathbf{x}_1) \le f_j(\mathbf{x}_2), \forall j \in \{1, 2\}$) and $\mathbf{x}_1$ is better than $\mathbf{x}_2$ on at least one objective (i.e., $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2), \exists j \in \{1, 2\}$).

Pareto-optimal solution: Solution $\mathbf{x}^* \in \{1, 2, \ldots, m \cdot p\}^n$ is regarded as Pareto-optimal if there exist no feasible solution dominating it.

Pareto Set/Front: All the Pareto-optimal solutions are defined as Pareto-Set (PS) in the decision space and Pareto-Front (PF) in the objective space.

Algorithm design

Given a workflow scheduling solution, the importance of each workflow task varies greatly. For instance, adjusting the mapping from a critical task to a resource often successively affects the execution of many tasks, including its successors and other tasks being executed after these tasks and their successors. Whereas adjusting the mapping from a non-critical task to a resource may have no impact on the execution cost and makespan of the workflow. Also, the importance of each workflow task varies from solution to solution. Then, decision variables corresponding to different workflow tasks pose an imbalance feature to optimization objectives. To deal with the large-scale decision variables in cloud workflow scheduling, the VCAES incorporates a novel cooperative coevolution (CC) mechanism to dynamically measure the contributions of decision variables and adaptively allocate evolution opportunities for each group of decision variables based on their contributions. The proposed VCAES follows the framework of traditional multi-objective evolutionary optimization, including initialization, reproduction operator, and selection operator, as shown in Algorithm 1.

Algorithm 1 The overall framework of VCAES

```
Input: The problem in (8); population size $N$; memory length $l$; group
       size of decision variables $s$
Output: A final population $P$
1: $P \leftarrow$ Randomly generate a population
2: $\mathbf{M} \leftarrow \mathbf{O}_{l\times n}$
3: $V \leftarrow$ Generate a set of reference vectors
4: $g \leftarrow$ Initialize the number of generations for CC
5: while does not reach termination condition do
6:     $[P, K] \leftarrow AdaptiveCoEvolution(P, \mathbf{M}, V, g)$
7:     for $j = 1 \to l-1$ do
8:         for $i = 1 \to n$ do
9:             $\mathbf{M}(j, i) \leftarrow \mathbf{M}(j+1, i)$
10:        end for
11:    end for
12:    for $i = 1 \to n$ do
13:        $\mathbf{M}(l, i) \leftarrow K(i)$
14:    end for
15:    $Q \leftarrow$ Generate a new population
16:    $P \leftarrow$ EnvironmentalSelection($P \bigcup Q, N$)
17: end while
```

As illustrated in Algorithm 1, the inputs of the proposed VCAES are the multi-objective cloud workflow scheduling problem, the population size, memory length for recording the variable contributions, the number of decision variables in one group. Once the VCAES reaches the termination condition, it will output an up-to-date population.

In the initialization stage, one population is generated randomly (Line 1). Next, an $l \times n$ matrix $\mathbf{M}$ is initialized to collect the contribution of each variable over the past l iterations (Line 2). The element in row j and column i represents the contribution of the i-th variable in the previous j-th iteration. Also, a set of uniformly distributed reference vectors are initialized to assist in calculating variable contributions (Line 3). In addition, the number of iterations for cooperative co-evolution during each generation is initialized (Line 4). These iterations will be allocated to each group of variables in proportion to the overall contribution of variables in the corresponding group.

After the initialization stage, the VCAES enters the main loop. During each generation, the proposed adaptive cooperative coevolution mechanism is triggered to distribute the decision variables into many groups and allocate evolution opportunities to each group according to variable contributions (Line 6). Next, the memory matrix of variable contributions is updated (Lines 7–14). It is worth noting that the decision variables in the cloud workflow scheduling are related to each other. The VCAES generates a new population by evolving all variables in each generation (Line 15). After that, the non-dominated sorting and elitist-preserving method in NSGA-II [41] is employed to select an offspring population P from the combined population $P \bigcup Q$ (Line 16).

Before introducing the proposed adaptive cooperative co-evolution mechanism, we define and illustrate the variable contributions.

Suppose Q is a population that generated by evolving the decision variables in group $G(i)$ and fixing other decision variables, the contribution of each variable in $G(i)$ is defined as:

$$K(j) = \sum_{q \in Q} \mathrm{FI}(\boldsymbol{q}), \forall j \in G(i), \tag{9}$$

where $\mathrm{FI}(\boldsymbol{q})$ denotes the fitness improvement of solution $\boldsymbol{q}$.

For a set of reference vector V, the one associated with solution $\boldsymbol{q}$ is defined as $\boldsymbol{v}^* = \arg_{v \in V} \min \langle \boldsymbol{f}(\boldsymbol{q}), \boldsymbol{v} \rangle$, where $\langle \boldsymbol{f}(\boldsymbol{q}), \boldsymbol{v} \rangle$ represents the acute angle between two vectors. Suppose $\boldsymbol{p}$ is the associated solution of the reference vector $\boldsymbol{v}^*$, then the fitness improvement of solution $\boldsymbol{q}$ can be calculated as follows:

$$\mathrm{FI}(\boldsymbol{q}) = \max\{0, \mathrm{Fit}(\boldsymbol{p}, \boldsymbol{v}^*) - \mathrm{Fit}(\boldsymbol{q}, \boldsymbol{v}^*)\}, \tag{10}$$

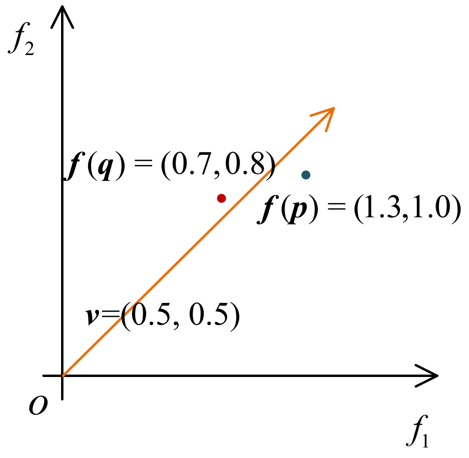
f_2
$\boldsymbol{f}(\boldsymbol{q}) = (0.7, 0.8)$
$\boldsymbol{f}(\boldsymbol{p}) = (1.3, 1.0)$
$\boldsymbol{v}=(0.5, 0.5)$
O
f_1

Fig. 2 An example workflow of seven tasks

where $\mathrm{Fit}(\boldsymbol{q}, \boldsymbol{v})$ denotes the fitness value of solution $\boldsymbol{q}$ with respect to the reference vector $\boldsymbol{v}$, which can be calculated as

$$\mathrm{Fit}(\boldsymbol{q}, \boldsymbol{v}) = \|\boldsymbol{f}'\| \cdot (\cos < \boldsymbol{f}', \boldsymbol{v} > + \sin < \boldsymbol{f}', \boldsymbol{v} >), \tag{11}$$

where $\boldsymbol{f}' = \boldsymbol{f}(\boldsymbol{q}) - \boldsymbol{z}$ denotes the translated objective vector, and $\boldsymbol{z}$ denotes the ideal point.

An intuitive example on calculating contribution is given in Fig. 2. Suppose previous solution $\boldsymbol{p}$ and the new solution $\boldsymbol{q}$ are associated to the reference vector $\boldsymbol{v} = (0.5, 0.5)$, and their objective vectors are $\boldsymbol{f}(\boldsymbol{p}) = (1.3, 1.0)$ and $\boldsymbol{f}(\boldsymbol{q}) = (0.7, 0.8)$, respectively. The ideal point is supposed to be $\mathbf{z} = (0, 0)$. Based on the definition in (11), we can obtain the fitness of these two solutions as $\mathrm{Fit}(\boldsymbol{p}, \boldsymbol{v}) = 1.8385$ and $\mathrm{Fit}(\boldsymbol{q}, \boldsymbol{v}) = 1.1314$. Then, the fitness improvement of solution $\boldsymbol{q}$ is $\mathrm{FI}(\boldsymbol{q}) = \max\{0, 1.8385 - 1.1314\} = 0.7071$. If the solution $\boldsymbol{q}$ is generated by evolving variables $\{1, 3, 5\}$, then the contribution of these variables are defined as $K(1) = K(3) = K(5) = 0.7071$.

Algorithm 2 gives the pseudo-code of the proposed adaptive cooperative coevolution mechanism, which dynamically distributes the variables with higher contributions into the same groups and assigns more evolution opportunities to accelerate the population convergence.

As illustrated in Algorithm 2, the inputs of the function *AdaptiveCoEvolution()* are the current population, the contribution of each variable during the previous l generations, the set of reference vectors, the population size, and the number of generations for cooperative coevolution search. Then, the outputs of this function are the updated population and the contribution of each variable.

At first, function *AdaptiveCoEvolution()* calculates the total contribution of each variable during the previous generations (Lines 1–4), where the notation $H(i)$ represents the total contribution of the i-th variable during the previous l generations. Then, it sorts the variables in a non-ascending

Algorithm 2 Function *AdaptiveCoEvolution()*

```
Input: Current population $P$; contribution matrix $\mathbf{M}$; reference vectors $V$; population size $N$; number of generations $g$
Output: A new population $P$; Contribution for each variable $K$
1: $H \leftarrow O_{1\times n}$
2: for $i = 1 \rightarrow n$ do
3:     $H(i) \leftarrow \sum_{j=1}^{l} \mathbf{M}(j,i) + \Delta$
4: end for
5: Sort variables by their contributions $H$ in a non-ascending order
6: $G \leftarrow$ Divide decision variables into $\lceil \frac{n}{s} \rceil$ groups
7: $K \leftarrow O_{1\times n}$
8: for $j = 1 \rightarrow | G |$ do
9:     $g' \leftarrow g \cdot \frac{\sum_{i\in G(j)} H(i)}{\sum_{i=1}^{n} H(i)}$
10:    for $k = 1 \rightarrow g'$ do
11:        $Q \leftarrow$ Generate an offspring population by evolving variables in $G(j)$
12:        $P \leftarrow$ EnvironmentalSelection($P \bigcup Q, N$)
13:    end for
14:    $c \leftarrow$ Calculate the contributions of variables in $G(j)$ as (9)
15:    for $i = 1 \rightarrow n$ do
16:        $K(i) \leftarrow \frac{c}{g'}$
17:    end for
18: end for
```

order according to their total contributions (Line 5) and distributes the variables into a series of groups (Line 6). In this way, the variables with similar contributions are distributed into the same group, which is helpful to distribute variables with larger contributions into the same groups to gain more evolution opportunities.

After that, function *AdaptiveCoEvolution()* successively evolves each group of variables, and measure their contributions. For a group of variables, the number of evolutionary iterations is calculated based on the sum of their contributions (Line 9). During each iteration, this function performs the reproduction operator on the corresponding variables to generate a new population (Line 11), and performs the selection operator to select the offspring population (Line 12). After evolving a group of variables, this function measures their contributions (Line 14). Also, it updates the contribution of the corresponding variables (Lines 15–17), where notation $K(i)$ represents the contribution of the i-th variable.

Regarding Function **AdaptiveCoEvolution**(), it costs $O(n \cdot l)$ to calculate the contribution of each decision variable (Lines 1–4, Algorithm 2). The time complexity of sorting decision variables is $O(n \log n)$ (Line 5, Algorithm 2). This function takes $O(N \cdot n)$ to reproduce an offspring population (Line 11, Algorithm 2). According to the analysis [41], the time complexity of the environmental selection is $O(N^2)$ (Line 12, Algorithm 2). Then, it takes $O(n \cdot (N \cdot n + N^2)) = O(n \cdot N \cdot (n + N)) = O(n^2 \cdot N)$ to adaptively each group of decision variables (Lines 8–18, Algorithm 2). Then, the time complexity of Function **AdaptiveCoEvolution**() is $O\left(n \cdot l + n \log n + n^2 \cdot N\right) = O(n^2 \cdot N)$, since l is often less than n.

Regarding the algorithm VCAES, the time complexity of updating the memory matrix is $O(n \cdot l)$ (Lines 7–11, Algorithm 1). The time complexities of reproducing an offspring population and selecting elitist solutions are $O(N \cdot n)$ and $O(N^2)$ (Lines 15-16, Algorithm 1), respectively. Thus, the time complexity of the VCAES during one generation is $O\left(n^2 \cdot N + n \cdot l + N \cdot n + N^2\right) = O(n^2 \cdot N)$.

Numerical experiments

To investigate the performance of the proposed VCAES, we compare it with four representative multi-objective cloud workflow scheduling algorithms: MOELS [42], EMS-C [9], WOF [43], LSMOF [44]. The MOELS and EMS-C follow the framework of NSGA-II [41] and incorporate new reproduction operators to generate offspring populations by evolving all the variables. The WOF and LSMOF are representative large-scale multi-objective evolutionary algorithms based on problem transformation.

Experimental setting

The eight types of workflows from different application domains, i.e., Inspiral (Gravitational physics), CyberShake (Earthquake), EpiGenomics (Biology), Montage (Astronomy), Sipht (Bioinformatics), BLAST (bioinformatics), Cycles (agroecosystems), and Seismology (seismology), have been widely used in evaluating cloud workflow scheduling algorithms. We select multiple task sizes from each workflow in the experiments. Besides, the DAG diagrams of some workflow instances with around 30 tasks are illustrated in Fig. 3. It is clear that these workflow instances cover various complicated structures, including in-tree, out-tree, fork-join, pipeline, and mixture. For more details on these workflows, please refer to the Pegasus repository.[3]

Five types of resources instances, i.e., t3.nano, t3.micro, t3.small, t3.medium, t3.medium, and t3.large, from Amazon EC2[4] are employed to simulate the cloud environments. We summarize the parameters of the five resource types in Table 1. Besides, we set the length of a billing period to 60 s and the bandwidth among resource instances to 5.0 Gbps.

Hypervolume [45] metric is designed to measure the quality of a population concerning both convergence and diversity, and has been frequently used to evaluate the performance of multi-objective evolutionary algorithms. Assume $\boldsymbol{r} = \{r_1, r_2\}$ is a reference point. The hypervolume value of a population P, corresponding to the volume between the reference point and the objective vectors of the solutions in

[3] https://confluence.pegasus.isi.edu/display/pegasus.

[4] https://aws.amazon.com/cn/ec2/pricing/on-demand/.

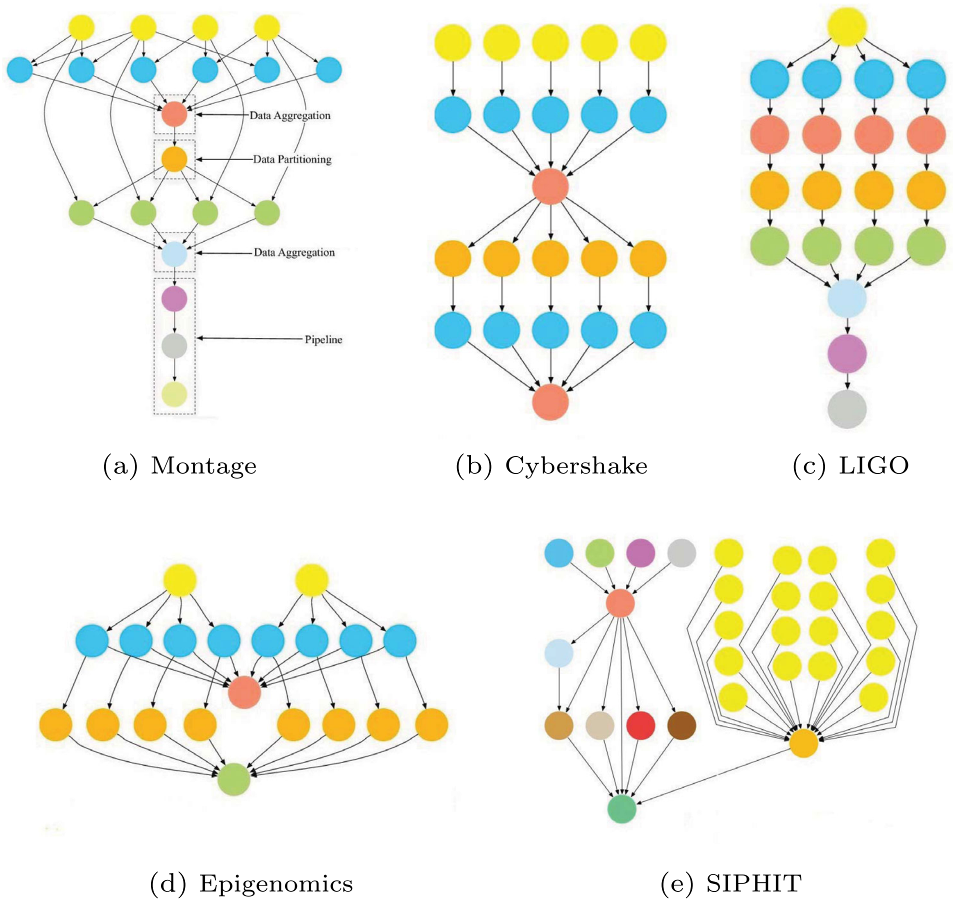

Fig. 3 DAG diagrams of workflows with about 30 tasks

Table 1 Parameters for five types of cloud resource

Type	Price ($/h)	vCPU	Memory (GB)
t3.nano	0.0062	2	0.5
t3.micro	0.0125	2	1.0
t3.small	0.025	2	2.0
t3.medium	0.0499	2	4.0
t3.large	0.0998	2	8.0

P, can be calculated as follows.

$$HV(P) = L\left(\bigcup_{p\in P}[f_1(p), r_1] \times [f_2(p), r_2]\right), \qquad (12)$$

where $L(\triangle)$ represents the Lebesgue measure.

Referring to the settings in MOELS and EMS-C, we set the population size of the five algorithms as 100. The maximum number of fitness evaluations is set as the stop condition for all the five algorithms and is set as 500,000.

The five algorithms are independently repeated 31 times on each workflow instance to mitigate random effects. We run all the experiments on a PC with two Cores i7-6500U CPU @ 2.50 GHz 2.59 GHz, 8.00 GB RAM, Windows 10.

Comparison result

Tables 2 and 3 report the average and standard deviation (in brackets) of the hypervolume values for the algorithms MOELS, EMS-C, WOF, LSMOF, and VCAES in scheduling the 27 workflow instances to cloud resources. For each workflow instance, the largest hypervolume value among the five algorithms is highlighted in bold. Besides, the Wilcoxon's rank-sum test with $\alpha = 0.05$ is used to examine the significant differences between the comparison algorithm and the proposed VCAES in terms of hypervolume metric. The marks $-$, $+$, and $\approx$ represent that the comparison algorithm is significantly worse than, better than, and similar to VCAES.

The comparison results in Tables 2 and 3 show that the proposed VCAES performs significantly better than the four baseline algorithms. Specifically, the proposal generates larger hypervolume values than MOELS, EMS-C, WOF, and LSMOF on 20, 22, 26, and 25 out of the 27 workflow instances respectively.

An interesting phenomenon is that the more decision variables, the more pronounced the advantages of the proposed VCAES. For example, in workflow instances with about 30 tasks, the VCAES obtains larger hypervolume values than algorithm LSMOF on 3 out of 5. Whereas, in workflow

Table 2 Comparison results for the five algorithms on 15 workflows in terms of the hypervolume metric

Workflows	*n*	MOELS	EMS-C	WOF	LSMOF	VCAES
CyberShake	50	8.8926e−1 −	8.8903e−1 −	8.6131e−1 −	**9.0899e−1** +	8.9113e−1
		(3.49e−3)	(2.83e−3)	(8.28e−3)	**(2.98e−3)**	(2.21e−3)
	100	9.3101e−1 −	9.3105e−1 −	9.1743e−1 −	9.2054e−1 −	**9.3232e−1**
		(4.43e−3)	(2.67e−3)	(6.42e−3)	(2.00e−4)	**(2.83e−3)**
	1000	9.2383e−1 −	9.2341e−1 −	9.1956e−1 −	9.1274e−1 −	**9.2764e−1**
		(3.19e−3)	(2.05e−3)	(7.85e−3)	(6.11e−4)	**(5.59e−3)**
Epigenomics	46	7.3227e−1 −	7.3273e−1 −	6.1543e−1 −	7.3249e−1 −	**7.3318e−1**
		(1.22e−3)	(8.57e−4)	(2.24e−3)	(2.95e−3)	**(6.68e−4)**
	100	7.4749e−1 ≈	7.4785e−1 ≈	5.1569e−1 −	7.4067e−1 −	**7.4792e−1**
		(3.61e−3)	(2.66e−3)	(9.38e−3)	(4.30e−3)	**(4.07e−3)**
	997	6.4607e−1 ≈	6.4565e−1 −	6.3796e−1 −	4.1166e−1 −	**6.4618e−1**
		(5.75e−3)	(5.45e−3)	(1.94e−3)	(2.43e−2)	**(4.61e−3)**
Inspiral	50	7.3357e−1 −	7.3415e−1 −	5.7999e−1 −	7.2639e−1 −	**7.3735e−1**
		(2.76e−3)	(2.91e−3)	(1.23e−3)	(3.23e−3)	**(1.64e−3)**
	100	7.1702e−1 −	7.1315e−1 −	6.1620e−1 −	6.8122e−1 −	**7.2373e−1**
		(4.29e−3)	(4.57e−3)	(1.02e−2)	(6.72e−3)	**(6.20e−3)**
	1000	6.0665e−1 ≈	6.0641e−1 ≈	**6.0958e−1** ≈	3.1438e−1 −	6.0681e−1
		(2.21e−3)	(4.00e−3)	**(5.40e−3)**	(1.87e−2)	(3.90e−3)
Montage	50	7.5877e−1 ≈	7.5419e−1 ≈	6.7955e−1 −	**7.7480e−1** +	7.6251e−1
		(5.51e−3)	(1.03e−2)	(2.96e−2)	**(7.20e−3)**	(1.02e−2)
	100	7.0910e−1 −	7.0575e−1 −	6.8191e−1 −	6.9895e−1 −	**7.1334e−1**
		(7.42e−3)	(9.27e−3)	(1.73e−2)	(9.76e−3)	**(7.98e−3)**
	1000	4.4897e−1 −	4.8142e−1 −	5.6040e−1 −	5.2911e−1 −	**5.7057e-1**
		(2.24e−1)	(2.13e−1)	(5.82e−3)	(4.40e−2)	**(1.77e−1)**
Sipht	50	6.1120e−1 −	6.1129e−1 −	5.2895e−1 −	6.1273e−1 −	**6.1469e−1**
		(4.47e−4)	(2.68e−4)	(2.47e−3)	(1.09e−4)	(3.34e−4)
	100	6.6995e−1 −	6.6972e−1 −	5.4832e−1 −	5.7389e−1 −	**6.7067e−1**
		(6.25e−4)	(6.01e−4)	(3.03e−3)	(5.76e−4)	**(9.73e−4)**
	1000	5.6588e−1 −	5.6763e−1 −	5.1232e−1 −	4.0689e−1 −	**5.7396e−1**
		(3.19e−3)	(5.44e−3)	(6.95e−2)	(3.10e−2)	**(4.47e−3)**

instances with 100 and 1000 tasks, the algorithm proposed in this paper significantly outperforms all the four comparison algorithms. Similar to the baseline algorithms MOELS and EMS-C, the proposed VCAES also follows the framework of NSGA-II. Different from MOELS and EMS-C, the VCAES integrates a new adaptive strategy based on variable contribution. Although two large-scale multi-objective evolutionary algorithms based on problem transformation, i.e., WOF and LSMOF, exhibit competitive performance in solving continuous problems, their performance in multi-objective scheduling workflows is far inferior to the proposed VCAES. The above comparison results demonstrate that the proposed adaptive mechanism in this paper can effectively improve the performance of evolutionary algorithms in solving multi-objective cloud workflow scheduling problems with large-scale variables.

To intuitively compare the convergence and diversity of the five multi-objective workflow scheduling algorithms, Fig. 4 illustrates the distribution of their output populations on workflow instances Inspiral, Montage, Sipht, and Cycles with large-scale tasks.

As illustrated in Fig. 4a, in the context of Inspiral with 100 tasks, the distribution of VCAES's output population is better than that of three baseline algorithms, i.e., MOELS, EMS-C, and WOF. More specifically, the output solutions of VCAES dominate that of the three algorithms. Although the diversity of solutions obtained by algorithm LSMOF is superior to that of the VCAES when the cost is less than 10, it is far inferior to algorithm VCAES during other range. To sum up, the proposed VCAES is superior to each baseline algorithm in terms of convergence and diversity. In the context of Montage with 100 tasks, the proposed VCAES has similar advantages

Table 3 Comparison results for the five algorithms on 15 workflows in terms of the hypervolume metric

Workflows	n	MOELS	EMS-C	WOF	LSMOF	VCAES
BLAST	103	6.6608e−1 −	6.6692e−1 −	6.2747e−1 −	5.9458e−1 −	**6.6844e−1**
		(3.21e−3)	(1.21e−3)	(6.54e−3)	(1.13e−2)	**(1.74e−3)**
	303	7.2511e−1 ≈	**7.2531e−1** +	7.2039e-1 −	4.3433e−1 −	7.2375e−1
		(1.55e−3)	**(1.59e−3)**	(1.93e−3)	(1.04e−2)	(1.89e−3)
Cycles	134	6.2794e−1	6.2718e−1 −	5.6875e−1 −	6.0116e−1 −	**6.3257e−1**
		(7.06e−3)	(6.64e−3)	(7.54e−3)	(7.40e−3)	**(5.95e−3)**
	219	5.5909e−1 −	5.5495e−1 −	5.2102e−1 −	4.9570e−1 −	**5.6454e−1**
		(4.12e−3)	(4.31e−3)	(9.82e−3)	(6.49e−3)	**(6.29e−3)**
	438	5.7681e−1 ≈	5.7292e−1 −	5.7542e−1 −	4.8407e−1 −	**5.7840e−1**
		(5.04e−3)	(3.60e−3)	(3.62e−3)	(1.04e−2)	**(4.99e−3)**
	657	4.8531e−1 −	4.7824e−1 −	4.8592e−1 −	3.6563e−1 −	**4.9047e−1**
		(7.96e−3)	(4.97e−3)	(5.93e−3)	(1.81e−2)	**(5.93e−3)**
	874	4.9787e−1 −	4.9839e−1 −	4.9265e−1 −	3.1602e−1 −	**5.0310e−1**
		(3.82e−3)	(4.17e−3)	(3.94e−3)	(1.32e−2)	**(3.16e−3)**
Seismology	101	8.0189e−1 −	8.0238e−1 −	7.5080e−1 −	7.9124e−1 −	**8.0762e−1**
		(4.66e−3)	(5.39e−3)	(1.02e−2)	(7.67e−3)	**(4.71e−3)**
	301	6.3724e−1 −	6.2538e−1 −	5.5509e−1 −	6.5746e−1 −	**6.8670e−1**
		(6.67e−3)	(6.39e−3)	(2.22e−2)	(4.20e−3)	**(3.75e−3)**
	501	6.0699e−1 −	6.0807e−1 −	5.2099e−1 −	5.9846e−1 −	**6.1478e−1**
		(5.44e−3)	(6.20e−3)	(3.07e−2)	(6.66e−3)	**(5.95e−3)**
	701	6.4371e−1 ≈	**6.4375e−1** ≈	3.6973e−1 −	6.2789e−1 −	6.4346e−1
		(5.11e−3)	**(5.34e−3)**	(6.34e−2)	(1.44e−2)	(4.57e−3)
	901	6.0329e−1 −	6.0284e−1 −	3.0147e−1 −	5.2011e−1 −	**6.3154e−1**
		(5.25e−3)	(6.72e−3)	(8.49e−2)	(1.25e−2)	**(5.78e−3)**

over the comparison algorithms in terms of convergence and diversity, as shown in Fig. 4b.

As can be observed from Fig. 4c, in the context of Sipht with 100 tasks, the solutions obtained by algorithm VCAES dominate most solutions obtained by comparison algorithms. This means that the VCAES outperforms all the four baseline algorithms in both convergence and diversity. In the context of Cycles with 657 tasks, the proposed VCAES has slight advantages over the comparison algorithms in terms of convergence and diversity, as shown in Fig. 4d.

Performance influence of different mechanisms

The VCAES mainly contains two new mechanisms to dynamically classify the decision variables and adaptively allocate evolution opportunities to each constructed group of decision variables. To measure the respective contributions of the two mechanisms to the overall performance, we construct three variants of the VCAES for comparison. The first variant, denoted as Variant 1, is constructed by replacing the decision variable classification mechanism with a random one and iterating each group of decision variables in a round-robin manner. The second variant, denoted as Variant 2, is constructed by replacing the decision variable classification mechanism with a random one and adaptively allocating evolution opportunities to each group of decision variables. The third variant, denoted as Variant 3, is constructed by removing the evolution opportunity allocation mechanism.

The main difference between the VCAES and its Variant 3 is that Variant 3 does not have the adaptive evolution opportunity allocation mechanism. Then, the improvement in the hypervolume value of the VCAES relative to Variant 3 can be attributed to the performance contribution of the adaptive evolution opportunity allocation mechanism. Similarly, the improvement in the hypervolume value of the VCAES relative to Variant 3 can be attributed to the performance contribution of the decision variable classification mechanism. The comparison results in Fig. 5 illustrate that the two proposed mechanisms contribute to the overall performance of the VCAES, with the decision variable classification mechanism contributing more. The performance improvement of the VCAES for Variant 1 can be attributed to the performance contribution of mixing the two proposed mechanisms. As shown in Fig. 5, in most workflow instances, the performance contribution of mixing the two mechanisms is better than that of any one. An exception is shown in Fig. 5b, where we can see that the overall performance contribution of mixing the two mechanisms is not as good as the contribution of the

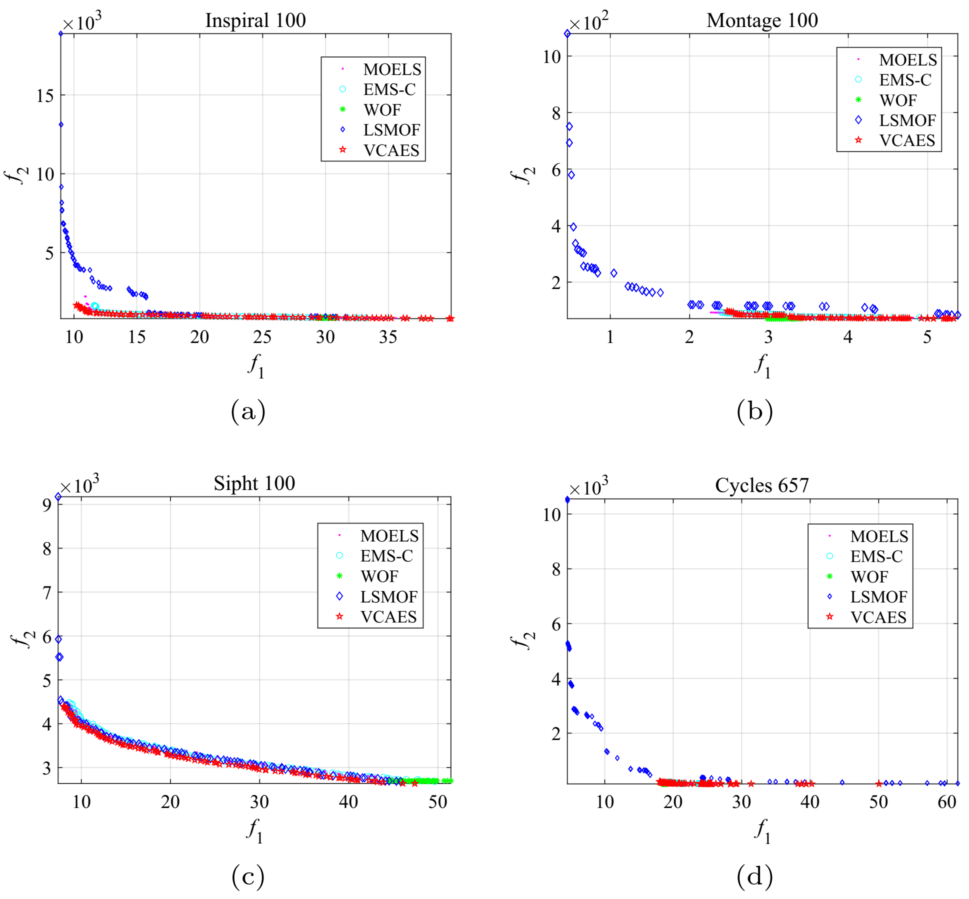

Fig. 4 Population distributions of the five algorithms on solving different workflow scheduling problems

decision variable classification mechanism. This is because a mechanism cannot be efficient in any scenario. Instead, it has certain advantages in some scenarios and inevitably has its disadvantages in other scenarios.

Conclusions

This paper focuses on two challenges in multi-objective cloud workflow scheduling: (1) large-scale decision variables; (2) and imbalance feature among variables regarding their contributions to objectives. To deal with these two challenges, this paper suggests a variable-contribution-based adaptive evolutionary cloud workflow scheduling approach that dynamically classifies the variables and adaptively allocates evolution opportunities to each constructed group of variables. Finally, in the context of real-world workflows and cloud platforms, this paper conducts comparison experiments to verify the effectiveness of the proposed adaptive mechanism in enhancing the population to approximate the Pareto-fronts of multi-objective cloud workflow scheduling problems.

Cloud workflow scheduling is a representative grey-box problem, and it is interesting to mine the knowledge on the workflows and cloud resources to derive efficient scheduling algorithms. Another potential direction is to design a parallel evolutionary framework to shorten the time overhead of evolution optimization to support cloud workflow scheduling in real-time and uncertain situations.

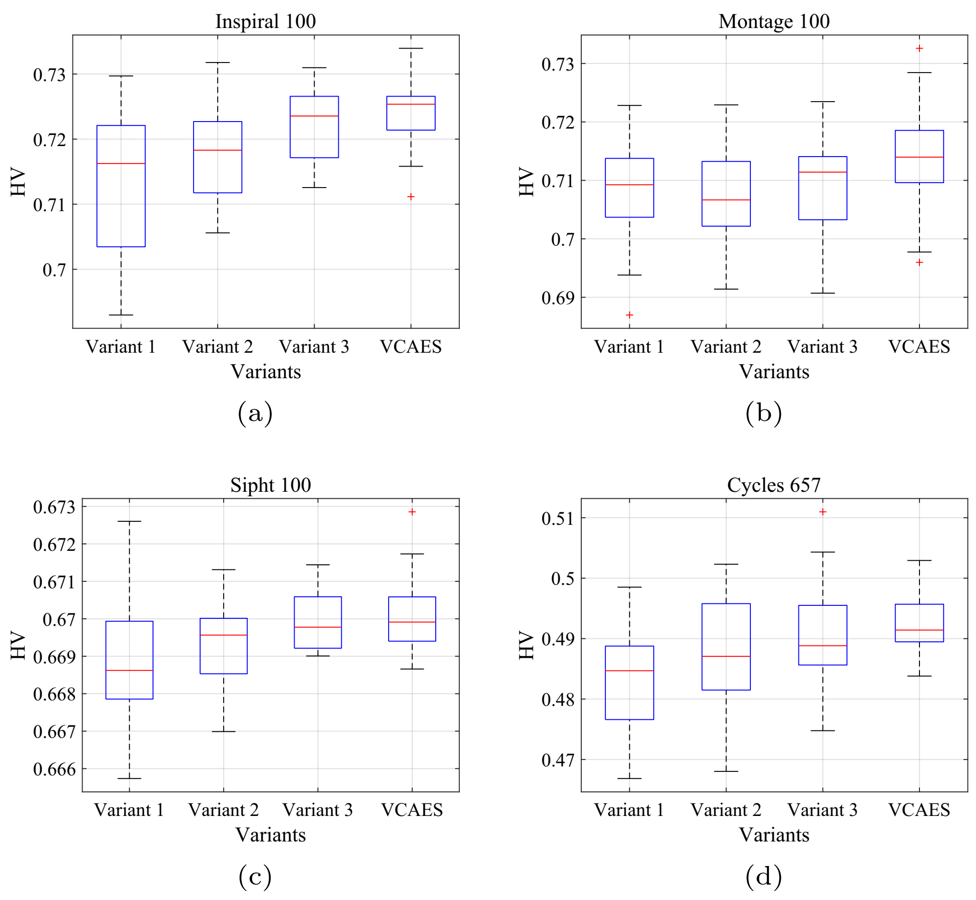

Fig. 5 Performance influence of the two proposed components

Acknowledgements This work is supported by the Science and Technology Innovation Team of Shaanxi Province (2023-CX-TD-07), and the Special Project in Major Fields of Guangdong Universities (2021ZDZX1019), the Major Projects of Guangdong Education Department for Foundation Research and Applied Research (2017KZDXM081, 2018KZDXM066), Guangdong Provincial University Innovation Team Project (2020KCXTD045) and the Hunan Key Laboratory of Intelligent Decision-making Technology for Emergency Management (2020TP1013).

Funding The funding has been received from National Natural Science Foundation of China with Grant no. 61773120; Hunan Provincial Innovation Foundation for Postgraduate with Grant no. CX20200585.

Data availability Data available on request from the authors.

References

1. Bugingo E, Zhang D, Chen Z, Zheng W (2021) Towards decomposition based multi-objective workflow scheduling for big data processing in clouds. Clust Comput 24(1):115–139
2. Lv Z, Lou R, Li J, Singh AK, Song H (2021) Big data analytics for 6G-enabled massive internet of things. IEEE Internet Things J 8(7):5350–5359
3. Lv Z, Qiao L, Hossain MS, Choi BJ (2021) Analysis of using blockchain to protect the privacy of drone big data. IEEE Netw 35(1):44–49
4. Cong P, Li L, Zhou J, Cao K, Wei T, Chen M, Hu S (2018) Developing user perceived value based pricing models for cloud markets. IEEE Trans Parallel Distrib Syst 29(12):2742–2756
5. Wang S, Sheng H, Zhang Y, Yang D, Shen J, Chen R (2023) Blockchain-empowered distributed multi-camera multi-target tracking in edge computing. IEEE Trans Ind Inf 2022:896
6. Farid M, Latip R, Hussin M, Hamid NAWA (2020) Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. IEEE Access 8:24309–24322
7. Masdari M, ValiKardan S, Shahi Z, Azar SI (2016) Towards workflow scheduling in cloud computing: a comprehensive analysis. J Netw Comput Appl 66:64–82
8. Cao B, Sun Z, Zhang J, Gu Y (2021) Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing. IEEE Trans Intell Transp Syst 22(6):3832–3840
9. Zhu Z, Zhang G, Li M, Liu X (2016) Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans Parallel Distrib Syst 27(5):1344–1357

10. Hosseinzadeh M, Ghafour MY, Hama HK, Vo B, Khoshnevis A (2020) Multi-objective task and workflow scheduling approaches in cloud computing: a comprehensive review. J Grid Comput 18(3):327–356
11. Xiao Z, Shu J, Jiang H, Lui JC, Min G, Liu J, Dustdar S (2022) Multi-objective parallel task offloading and content caching in D2D-aided MEC networks. IEEE Trans Mob Comput 2022:896
12. Cao B, Yan Y, Wang Y, Liu X, Lin JC-W, Sangaiah AK, Lv Z (2022) A multiobjective intelligent decision-making method for multi-stage placement of PMU in power grid enterprises. IEEE Trans Ind Inf 2022:87
13. Zhan Z-H, Liu X-F, Gong Y-J, Zhang J, Chung HS-H, Li Y (2015) Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput Surv 47(4):1–33
14. Durillo JJ, Nae V, Prodan R (2014) Multi-objective energy-efficient workflow scheduling using list-based heuristics. Futur Gener Comput Syst 36:221–236
15. Fard HM, Prodan R, Fahringer T (2014) Multi-objective list scheduling of workflow applications in distributed computing infrastructures. J Parall Distrib Comput 74(3):2152–2165
16. Han P, Du C, Chen J, Ling F, Du X (2021) Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. J Syst Architect 112:101837
17. Ismayilov G, Topcuoglu HR (2020) Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. Futur Gener Comput Syst 102:307–322
18. Li X, Sun Y (2021) Application of RBF neural network optimal segmentation algorithm in credit rating. Neural Comput Appl 33:8227–8235
19. Qin X, Liu Z, Liu Y, Liu S, Yang B, Yin L, Liu M, Zheng W (2022) User OCEAN personality model construction method using a BP neural network. Electronics 11(19):3022
20. 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 (2019) Multiobjective cloud workflow scheduling: a multiple populations ant colony system approach. IEEE Trans Cybern 49(8):2912–2926
21. Adhikari M, Amgoth T, Srirama SN (2020) Multi-objective scheduling strategy for scientific workflows in cloud environment: a firefly-based approach. Appl Soft Comput 93:106411
22. Gupta R, Gajera V, Jana PK et al (2016) An effective multi-objective workflow scheduling in cloud computing: a PSO based approach. In: 2016 Ninth International Conference on Contemporary Computing, pp 1–6, IEEE
23. Yu H (2021) Evaluation of cloud computing resource scheduling based on improved optimization algorithm. Compl Intell Syst 7(4):1817–1822
24. Wang Y, Zuo X (2021) An effective cloud workflow scheduling approach combining PSO and idle time slot-aware rules. IEEE/CAA J Autom Sin 8(5):1079–1094
25. Abed-Alguni BH, Alawad NA (2021) Distributed grey wolf optimizer for scheduling of workflow applications in cloud environments. Appl Soft Comput 102:107113
26. Choudhary A, Gupta I, Singh V, Jana PK (2018) A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing. Futur Gener Comput Syst 83:14–26
27. Hosseini Shirvani M, Noorian Talouki R (2022) Bi-objective scheduling algorithm for scientific workflows on cloud computing platform with makespan and monetary cost minimization approach. Compl Intell Syst 8(2):1085–1114
28. Mohammadzadeh A, Masdari M, Gharehchopogh FS (2021) Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm. J Netw Syst Manage 29(3):1–34
29. Zhang H, Wu Y, Sun Z (2022) EHEFT-R: multi-objective task scheduling scheme in cloud computing. Compl Intell Syst 8(6):4475–4482
30. Zhou X, Zhang G, Sun J, Zhou J, Wei T, Hu S (2019) Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. Futur Gener Comput Syst 93:278–289
31. Kumar MS, Tomar A, Jana PK (2021) Multi-objective workflow scheduling scheme: a multi-criteria decision making approach. J Ambient Intell Hum Comput 2021:1–20
32. Ye X, Liu S, Yin Y, Jin Y (2017) User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm. Knowl-Based Syst 135:113–124
33. Pham T-P, Fahringer T (2020) Evolutionary multi-objective workflow scheduling for volatile resources in the cloud. IEEE Trans Cloud Comput 2020:36
34. Tian Y, Si L, Zhang X, Cheng R, He C, Tan KC, Jin Y (2021) Evolutionary large-scale multi-objective optimization: a survey. ACM Comput Surv 54(8):1–34
35. Coello CAC, Brambila SG, Gamboa JF, Tapia MGC, Gómez RH (2020) Evolutionary multiobjective optimization: open research areas and some challenges lying ahead. Compl Intell Syst 6(2):221–236
36. Chen H, Cheng R, Wen J, Li H, Weng J (2020) Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. Inf Sci 509:457–469
37. Zhang X, Tian Y, Cheng R, Jin Y (2018) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Trans Evol Comput 22(1):97–112
38. Chen H, Zhu X, Liu G, Pedrycz W (2021) Uncertainty-aware online scheduling for real-time workflows in cloud service environment. IEEE Trans Serv Comput 14(4):1167–1178
39. De Maio V, Kimovski D (2020) Multi-objective scheduling of extreme data scientific workflows in fog. Futur Gener Comput Syst 106:171–184
40. Lv Z, Xiu W (2019) Interaction of edge-cloud computing based on SDN and NFV for next generation IoT. IEEE Internet Things J 7(7):5706–5712
41. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
42. Wu Q, Zhou M, Zhu Q, Xia Y, Wen J (2020) MOELS: multiobjective evolutionary list scheduling for cloud workflows. IEEE Trans Autom Sci Eng 17(1):166–176
43. Zille H, Ishibuchi H, Mostaghim S, Nojima Y (2018) A framework for large-scale multiobjective optimization based on problem transformation. IEEE Trans Evol Comput 22(2):260–275
44. He C, Li L, Tian Y, Zhang X, Cheng R, Jin Y, Yao X (2019) Accelerating large-scale multiobjective optimization via problem reformulation. IEEE Trans Evol Comput 23(6):949–961
45. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Trans Evol Comput 3(4):257–271