mathematics

MDPI

Article

Knowledge-Based Evolutionary Optimizing Makespan and Cost for Cloud Workflows

Lining Xing [1], Rui Wu [2], Jiaxing Chen [2] and Jun Li [3,*]

[1] School of Electronic Engineering, Xidian University, Xi'an 710071, China
[2] Inner Mongolia Institute of Dynamical Machinery, Hohhot 010010, China
[3] School of Management, Hunan Institute of Engineering, Xiangtan 411104, China
* Correspondence: 08036@hnie.edu.cn

Abstract: Workflow scheduling is essential to simultaneously optimize the makespan and economic cost for cloud services and has attracted intensive interest. Most of the existing multi-objective cloud workflow scheduling algorithms regard the focused problems as black-boxes and design evolutionary operators to perform random searches, which are inefficient in dealing with the elasticity and heterogeneity of cloud resources as well as complex workflow structures. This study explores the characteristics of cloud resources and workflow structures to design a knowledge-based evolutionary optimization operator, named KEOO, with two novel features. First, we develop a task consolidation mechanism to reduce the number of cloud resources used, reducing the economic cost of workflow execution without delaying its finish time. Then, we develop a critical task adjustment mechanism to selectively move the critical predecessors of some tasks to the same resources to eliminate the data transmission overhead between them, striving to improve the economic cost and finish time simultaneously. At last, we embed the proposed KEOO into four classical multi-objective algorithms, i.e., NSGA-II, HypE, MOEA/D, and RVEA, forming four variants: KEOO-NSGA-II, KEOO-HypE, KEOO-MOEA/D, and KEOO-RVEA, for comparative experiments. The comparison results demonstrate the effectiveness of the KEOO in improving these four algorithms in solving cloud workflow scheduling problems.

Keywords: evolutionary computation; workflow scheduling; cloud computing; multi-objective optimization; evolutionary operator

MSC: 97M40; 97P30

Citation: Xing, L.; Wu, R.; Chen, J.; Li, J. Knowledge-Based Evolutionary Optimizing Makespan and Cost for Cloud Workflows. *Mathematics* **2023**, *11*, 38. https://doi.org/10.3390/math11010038

Academic Editors: Linqiang Pan, Zhihua Cui, Harish Garg, Thomas Hanne and Gai-Ge Wang

Received: 1 November 2022
Revised: 6 December 2022
Accepted: 19 December 2022
Published: 22 December 2022

1. Introduction

Big data processing applications from various domains, e.g., the Earthquake and Internet of Things, can be divided into a series of phases. In addition, tasks belonging to different phases have complex data dependencies. These applications are commonly described as workflows [1,2]. Due to the substantial computation and data transmission requirements, executing these workflows often requires massive high-performance infrastructures. With the benefits of pay-per-use, elasticity, scalability, high reliability, and flexibility, cloud computing has become an increasingly attractive choice for enterprises to process their workflow applications by alleviating the burden of building, operating, and maintaining infrastructure [3–5].

Scheduling workflows in clouds, which determines the mappings from tasks to resources and the task order on each resource, is of paramount importance to optimize the execution makespan and economic cost, satisfying the quality of service for cloud users and earning more profits for cloud providers [6]. Since the cloud workflow scheduling problem is NP-complete [7,8] and involves multiple conflicting objectives [9], many studies choose evolutionary optimization techniques to obtain satisfactory solutions within an acceptable

time [10]. Most existing studies adopt multi-objective optimization for cloud workflow scheduling by designing new selection or reproduction operators.

Some studies have tried to design selection operators for multi-objective cloud workflow scheduling. For instance, Zhou et al. [11] combined a fuzzy-dominance-sort-based selection operator with the earliest-finish-time-based reproduction operator to balance makespan and economic cost for workflow execution in clouds. Kumar et al. [12] integrated the entropy weight approach into a multi-criteria decision-making technique to optimize makespan, economic cost, energy consumption, and reliability. Ye et al. [13] enhanced knee point-driven evolutionary algorithm to balance makespan, the average execution time of all workflow tasks, reliability, and economic cost of workflow execution. Pham et al. [14] considered the volatility of spot cloud instances and employed a multi-objective evolutionary algorithm to balance makespan and economic cost for workflow execution in clouds.

Compared with the selection operators, designing problem-specific reproduction operators attracts more interest. Up to now, considerable efforts have been devoted to developing new reproduction operators for multi-objective workflow scheduling in cloud computing [10,15,16]. At first, some works [17–19] replaced the reproduction operator of the multi-objective optimization framework with the list-based heuristic rule for cloud workflow scheduling. For instance, Durillo and Prodan [17] improved a task list-based heuristic rule to obtain a series of intermediate non-dominated solutions for each workflow task, and employed the fast non-dominated sorting-based approach [20] to maintain a predefined number of non-dominated solutions. Wu et al. [21] designed a task list-based optimization rule and a preference weight evolutionary strategy to balance the economic cost and makespan of cloud workflows. Although the heuristics-based multi-objective optimization algorithms pose low time overheads and are effective in specific scenarios, their global search capacity is poor, especially in the face of complex and diverse workflows.

Then, bio-inspired optimization techniques, such as ant colony optimization [22], particle swarm optimization [23,24], artificial neural network [25], genetic algorithm [26,27], and grey wolf optimization [28], were adopted to improve the capacity of reproduction operators in multi-objective evolutionary algorithms. For instance, Zhu et al. [9] improved the multi-objective evolutionary algorithm with problem-specific encoding, population initialization, and reproduction operators to optimize both the makespan and economic cost of cloud workflows. Chen et al. [22] designed a particle swarm optimization algorithm with two colonies to balance the makespan and economic cost of workflow execution in clouds. Ismayilov et al. [25] incorporated an artificial neural network into the NSGA-II to optimize six objectives of workflow execution in cloud computing. Wang et al. [24] embedded idle time gap-based strategies into particle swarm optimization to optimize the economic cost of workflows. In addition, some works integrated heuristic rules and bio-inspired algorithms to reproduce the new population. For instance, Choudhary et al. [29] suggested a hybridization of gravitational search algorithm and heterogeneous-earliest-finish-time for bi-objective scheduling cloud workflows. Mohammadzadeh et al. [30] suggested a hybridization of the antlion and grasshopper optimization algorithm to optimize makespan, economic cost, energy consumption, and throughput of workflow execution in clouds. These existing multi-objective cloud workflow scheduling algorithms often regard the focused problems as black-boxes, and search the solutions in a random way, resulting in low efficiency in dealing with the elasticity and heterogeneity of cloud resources as well as complex workflow structures.

By analyzing these existing works, we can derive that workflow scheduling challenges the randomness of evolutionary search from two aspects. On the one hand, the elasticity, heterogeneity, and on-demand use of cloud computing resources provide a vast number of candidate resources for each workflow task, meaning that the search space for scheduling a cloud workflow is expanding explosively. It is inefficient for the multi-objective evolutionary algorithms to search explosive growth space randomly. On the other hand, due to data dependency among workflow tasks, randomly adjusting a task's execution scheme

often successively affects a series of tasks, including its successor tasks and those tasks being executed after these tasks and their successor tasks.

The current challenges of scheduling cloud workflows motivate us to explore the knowledge of cloud resources and workflow structures to design an effective multi-objective evolutionary optimization algorithm. More specifically, we explore the heterogeneity, pay-as-you-go, and elasticity of cloud resources to design a consolidation mechanism to merge workflow tasks on different cloud resources without delaying any tasks. This way helps reduce the economic cost by reducing the number of cloud resources used and improve search efficiency by shrinking the set of candidate cloud resources. Moreover, the knowledge that data transmission overheads among tasks on the same resource are negligible motivates us to design a critical task adjustment mechanism. It selectively moves the critical predecessors of some tasks to the same resources to eliminate the data transmission overhead between them, striving to improve the finish time and economic cost simultaneously. At last, based on real-world workflows and cloud platforms, we conduct comparative experiments to demonstrate that the proposed approach is capable of improving the performance of multi-objective evolutionary algorithms in solving cloud workflow scheduling problems.

This paper is organized as follows. Section 2 mathematically formulates the multi-objective workflow scheduling problem. Section 3 describes the proposed KEOO, followed by experimental verifications in Section 4. Section 5 concludes this paper.

2. Problem Formulation

This section provides the models for workflow and cloud resource, then formulates the multi-objective cloud workflow scheduling problem.

2.1. Workflow Model

Generally, the workflow structure is modeled as a Directed Acyclic Graph (DAG), in which the nodes and directed edges denote the tasks and the data dependencies among the tasks, respectively. In detail, the DAG model of a workflow is formulated as $\Psi = \{T, D\}$, where $T = \{t_1, t_2, \cdots, t_n\}$ is the set of nodes representing the workflow tasks, and $D \subseteq T \times T$ is the set of edges representing data dependencies among the tasks. The existence of edge $d_{i,j} \in D$ means that the start of task t_j requires the output result of task t_i as input. Then, task t_i is regarded as an immediate predecessor of task t_j, and t_j is regarded as an immediate successor of t_i. For a task t_i, the set of all its immediate predecessors is represented as $P(t_i)$, while the set of all its immediate successors is represented as $S(t_i)$.

Figure 1 provides an intuitive example of a DAG model for a workflow having seven tasks, i.e., $T = \{t_1, t_2, \cdots, t_7\}$. The edge $d_{1,2}$ represents the data dependency between t_1 and t_2, meaning that the start of task t_2 needs to wait for the output result of task t_1. As can be seen in Figure 1, for task t_7, the set of its immediate predecessors is $P(t_7) = \{t_5, t_6\}$, and the set of t_6's immediate successors is $S(t_6) = \{t_7\}$.

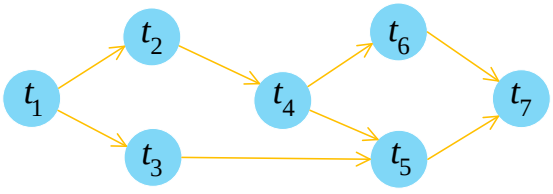


Figure 1. DAG model of a workflow with seven tasks.

2.2. Cloud Resource Model

Infrastructure as a Service (IaaS) is one popular cloud paradigm, where cloud providers provide unlimited cloud resources with various types [31]. Different resource types differ in price and performance configurations, such as CPU frequency, network bandwidth,

memory, and storage size. Given that the cloud platform provides m types of resources, we describe all these resource types as $\Gamma = \{1, 2, \cdots, m\}$, where $\tau \in \Gamma$ corresponds to the τ-th resource type. For a resource type τ, its price and configurations are respectively represented as $pr(\tau)$ and $con(\tau)$. Then, a resource instance of type τ in a cloud platform can be modeled as $r_k^\tau = \{k, pr(\tau), con(\tau)\}$, where k denotes the index of this resource instance.

We refer to well-known cloud providers, e.g., Amazon EC2 and Alibaba Cloud ECS, and follow their resource charging mode of pay-as-you-use. This way, each user can rent cloud instances on-demand and pay for the used instances based on the real usage time. Generally, the cloud providers charge for resource instances according to the number of charging periods and round up the partial time of a period to one more period. If the period length is one hour, the number of charging periods for 60.5 min is two.

The network structure among resource instances in clouds is often heterogeneous and intricate. Since this paper focuses on scheduling workflow tasks, we simplify the underlying network structure and assume that all resource instances are interconnected. The symbol $b_{k,l}$ is employed to denote the communication bandwidth between resource instance r_k^τ and $r_l^{\tau'}$. When two data-dependent tasks are executed on the same resource instance, they will use the same storage space, and there is no data transmission through the network. Then, the data transmission overhead between these two tasks can be negligible [9,32].

2.3. Multi-Objective Scheduling Cloud Workflows

Since cloud resource instances are elastic, we construct a resource pool based on the workflow's most used resource instances. We use p to represent the maximum parallelism of the workflow, and then the resource pool contains p instances of each type. That is to say, the resource pool can be detailed as

$$R = \{r_1^1, r_2^1, \cdots, r_p^1, r_{p+1}^2, r_{p+1}^2, \cdots, r_{2\cdot p}^2, \cdots, r_{m\cdot p}^m\}. \tag{1}$$

This paper defines the decision vector $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ as the mappings from workflow tasks to resource instances, where the i-th decision variable x_i corresponds to the i-th workflow task and its value indicates the index of this task mapping resource instance. Then, the value of each decision variable is one of the elements of the set $\{1, 2, \cdots, m \cdot p\}$.

For a decision vector, we assume that the workflow task t_i is mapped to resource instance r_k^τ. This task's start time $st_{i,k}$ is the maximum time to collect the output results from all its predecessors and the available mapped resource instance.

On resource instance r_k^τ, the task set ahead of task t_i is described as:

$$B_i = \{t_p | O(t_p) < O(t_i)\}, \tag{2}$$

where $O(t_p)$ denotes the order number of task t_p on the resource instance r_k^τ.

Then, the start time $st_{i,k}$ of workflow task t_i on the mapped resource instance r_k^τ is calculated as follows:

$$st_{i,k} = \max\{\max_{t_b \in B_i} ft_{b,k}, \max_{t_p \in P(t_i)} \{ft_{p,*} + dt_{p,i}\}\}, \tag{3}$$

where $ft_{b,k}$ denotes the finish time of task t_b on resource instance r_k^τ, $ft_{p,*}$ denotes the finish time of task t_p on its mapped resource instance, and $dt_{p,i}$ denotes the data transmission time from t_p to t_i.

Before scheduling, the execution time $et_{i,k}$ of workflow task t_i on resource instance r_k^τ can be estimated by the computation amount of the task and the performance configuration of the resource instance. Then, the relationships among $st_{i,k}$, $et_{i,k}$, and $ft_{i,k}$ can be described as follows:

$$ft_{i,k} = st_{i,k} + et_{i,k}. \tag{4}$$

The data dependencies among workflow tasks mean that a task t_i cannot start execution before receiving the output data from all its predecessors, which creates the following constraint:

$$st_{i,k} \geq \max_{t_p \in P(t_i)} \{ft_{p,r(t_p)} + \frac{I\{r(t_i) \neq r(t_p)\} \times w(e_{p,i})}{bw}\}, \ \forall t_i \in T, \tag{5}$$

where $I\{\cdot\}$ is an indicator function. If t_p and t_i are mapped to different resources, $I\{\cdot\}$ is 1; otherwise, it is 0. The indicator function is employed to reflect the fact that once two dependent tasks are executed by the same resource, the data transmission overhead between these two tasks is negligible and assumed to be zero. bw denotes the bandwidth.

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

$$T_k = \{t_i | x_i = k, i \in \{1, 2, \cdots, n\}\}. \tag{6}$$

With the mapped task set T_k, the startup time ut_k and shutdown time nt_k of resource instance r_k^τ can be calculated as follows:

$$\begin{aligned} ut_k &= \min_{t_i \in T_k} \{st_{i,k} - \max_{t_p \in P(t_i)} dt_{p,i}\}, \\ nt_k &= \max_{t_i \in T_k} \{ft_{i,k} + \max_{t_s \in S(t_i)} dt_{i,s}\}. \end{aligned} \tag{7}$$

With the formulation above, we formulate the first optimization objective, i.e., minimizing the economic cost, as follows:

$$\text{Min } f_1(\boldsymbol{x}) = \sum_{r_k^\tau \in R} pr(\tau) \times \lceil \frac{nt_k - ut_k}{C} \rceil, \tag{8}$$

where C denotes the length of charging period for resource instances.

The second optimization objective is to minimize the makespan of the workflow, which refers to the maximum finish time of all the workflow tasks. We formulate this optimization objective as follows:

$$\text{Min } f_2(\boldsymbol{x}) = \max_{t_i \in T} ft_{i,*}. \tag{9}$$

To summarize, the model for multi-objective scheduling cloud workflows can be formulated as follows:

$$\begin{cases} \text{Min} & \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})], \\ \text{S.t.} & \\ & \boldsymbol{x} \in \{1, 2, \cdots, m \cdot p\}^n, \\ & st_{i,k} \geq \max_{t_p \in P(t_i)} \{ft_{p,r(t_p)} + \frac{I\{r(t_i) \neq r(t_p)\} \times w(e_{p,i})}{bw}\}, \ \forall t_i \in T. \end{cases} \tag{10}$$

To improve readability, we take the workflow in Figure 1 as an example to visually illustrate the decision variable and the calculation of the corresponding objective vector. Assuming that the resource set is $R = \{r_1, r_2, \cdots, r_5\}$, one decision variable of the workflow with seven tasks in Figure 1 is $\mathbf{x} = \{2, 1, 3, 2, 4, 5, 2\}$. The configurations of the five cloud resources and the execution time from tasks to resources are summarized in Table 1. The data transmission time among workflow tasks is given in Table 2. Based on the above assumptions, Figure 2 illustrates the Gantt chart of the schedule. Then, we can calculate each workflow task's start and finish time, which is given in Table 3.

Table 1. Examples of task execution time (in minute).

	r_1	r_2	r_3	r_4	r_5
CPU (GHz)	2.9	3.3	2.9	2.9	3.3
Memory Size (GB)	4.0	6.0	4.0	4.0	6.0
Bandwidth (MB/s)	20.0	30.0	20.0	20.0	30.0
Storage (GB)	500.0	1000.0	500.0	500.0	1000.0
Price ($/h)	0.025	0.032	0.025	0.025	0.032
t_1	6.25	5.0	6.25	6.25	5.0
t_2	5.0	4.0	5.0	5.0	4.0
t_3	10.0	8.0	10.0	10.0	8.0
t_4	6.25	5.0	6.25	6.25	5.0
t_5	15.0	12.0	15.0	15.0	12.0
t_6	25.0	20.0	25.0	25.0	20.0
t_7	22.5	18.0	22.5	22.5	18.0

Table 2. Examples of data transmission time (in minutes) among tasks.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
t_1	–	3	3	–	–	–	–
t_2	–	–	–	2	–	–	–
t_3	–	–	–	–	5	–	–
t_4	–	–	–	–	3	5	–
t_5	–	–	–	–	–	–	10
t_6	–	–	–	–	–	–	3
t_7	–	–	–	–	–	–	–

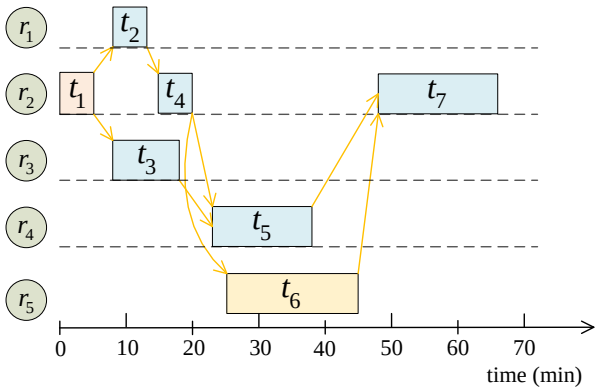


Figure 2. Example of a schedule.

Table 3. Start time $st(\cdot)$ and finish time $ft(\cdot)$ (in minute) of each workflow task.

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
$st(\cdot)$	0.0	8.0	8.0	15.0	23.0	25.0	48.0
$ft(\cdot)$	5.0	13.0	18.0	20.0	38.0	45.0	66.0

According to the charging mode of cloud resources, i.e., the partial time of a charging period is rounded up to one, the charging periods of the five resources are 1, 2, 1, 1, and 1, respectively. Then, the execution cost of the workflow is $0.025 \times 1 + 0.032 \times 2 + 0.025 \times 1 + 0.025 \times 1 + 0.032 \times 1 = 0.171$ dollars, which corresponds to the first optimization objective. Besides, the makespan of a workflow refers to the maximum finish time of all the tasks. We can obtain this optimization objective as $\max\{5.0, 13.0, 18.0, 20.0, 38.0, 45.0, 66.0\} = 66.0$ min.

Pareto dominance is commonly used to compare solutions with multiple conflicting objectives [33,34].

Pareto-Dominance: Suppose $\mathbf{x}_1$ and $\mathbf{x}_2$ are two feasible solutions for the cloud workflow scheduling. $\mathbf{x}_1$ is regarded to Pareto dominate $\mathbf{x}_2$ (expressed as $\mathbf{x}_1 \prec \mathbf{x}_2$) if and only if the two objectives of $\mathbf{x}_1$ is no larger than that of $\mathbf{x}_2$ (i.e., $f_j(\mathbf{x}_1) \leq f_j(\mathbf{x}_2), \forall j \in \{1,2\}$) and $\mathbf{x}_1$ is less 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: A solution $\mathbf{x}^* \in \{1,2,\cdots,m \cdot p\}^n$ is generally defined as Pareto-optimal when there exists no feasible solution dominating it.

Pareto-optimal 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.

3. Algorithm Design

The framework of traditional multi-objective evolutionary algorithms includes initialization, reproduction operator, and selection operator [35]. The proposed KEOO contributes to strengthening reproduction operators' search capability by exploring the knowledge of cloud resources and workflow structures. It incorporates a task consolidation mechanism to efficiently search explosive growth solution space caused by the heterogeneity and elasticity of cloud resources, and a critical task adjustment mechanism to handle the complex workflow structures. Algorithm 1 provides the overall framework of a multi-objective evolutionary algorithm embracing the proposed KEOO.

Algorithm 1: The main framework of KEOO

Input: The optimization problem in (10); population size N;
Output: A final population P;
1 $P \leftarrow$ Randomly generate a population;
2 **while** *does not reach termination condition* **do**
3 **if** *rand* < 0.5 **then**
4 $Q \leftarrow$ Reproduce an offspring population;
5 $Q \leftarrow$ Adjust Q using **TaskConsolidation()**;
6 **else**
7 $Q \leftarrow$ Reproduce an offspring population by **CriticalTaskReschedule()**;
8 $P \leftarrow$ EnvironmentalSelection($P \bigcup Q, N$);

As illustrated in Algorithm 1, the inputs of the proposed KEOO are the multi-objective cloud workflow scheduling problem and the population size. Once the KEOO reaches the termination condition, it outputs an up-to-date population.

In the initialization stage, one population is generated randomly (Line 1). Then, the KEOO iterates the reproduction and environmental selection stages until the termination condition is reached. During the reproduction stage, either the task consolidation mechanism (Line 5) or the critical task adjustment mechanism (Line 7) is triggered to reproduce an offspring population. Because the task consolidation mechanism is a heuristic rule and lacks global search capability, this mechanism is designed to further optimize the offspring solutions generated by traditional evolutionary algorithms (Line 4), rather than being used alone. Because the focused cloud workflow scheduling problem in this paper only optimizes two objectives of makespan and economic cost, many classical environmental selection operators are competent. Thus, this paper does not design a new selection operator but directly employs existing ones, such as NSGA-II [20], HypE [36], MOEA/D [37], and RVEA [38].

Algorithm 2 gives the pseudo-code of the task consolidation mechanism, which explores the heterogeneity, pay-as-you-go, and elasticity of cloud resources to consolidate workflow tasks to as few cloud resources as possible. Then, the number of leased resources can be reduced to achieve the goal of reducing the economic cost.

Algorithm 2: Function *TaskConsolidation()*

Input: Current population P;
Output: A new population P;

```
1  Sort R according to their CPU configurations;
2  k ← 1;
3  G_k ← {R_1};
4  conf ← CONF(R_1);
5  for j = 2 →| R | do
6      if CONF(R_j) == conf then
7          G_k ← G_k ⋃ {R_j};
8      else
9          k ← k + 1;
10         G_k ← {R_j};
11         conf ← CONF(R_j);
12 for h = 1 → k do
13     Sort G_h according to their completion time;
14     r ← G_h^1;
15     ct ← ct(G_h^1);
16     for j = 2 →| G_h | do
17         if st(G_h^j) ≥ ct then
18             for i = 1 → n do
19                 if x_i == I(G_h^j) then
20                     x_i ← I(r);
21         else
22             r ← G_h^j;
23         ct ← ct(G_h^j);
```

As illustrated in Algorithm 2, the function *TaskConsolidation()* consists of two phases: resource classification and task consolidation. To avoid affecting the task execution, the function only merges tasks on the resources with the same configurations. Then, during the phase of resource classification, all the resources are classified into a series of groups (Lines 1–11). At first, this function sorts all the resources according to their CPU configurations (Line 1). The symbol k is used to record the group number, initialized as 1 (Line 2). The set G_k is used to record the resources belonging to the k-th group (Line 3). The operator $CONF(\cdot)$ means to obtain the configuration information of a cloud resource. Next, each resource is checked (Line 5). If its configuration is consistent with the recorded one, it will be added to the current group (Line 7). Otherwise, a new group is created (Lines 9–10), and the recorded configuration is updated (Line 11).

After resource classification, the function *TaskConsolidation()* enters the task consolidation phase. It sorts a group of cloud resources in ascending order according to the maximum finish time of the mapped tasks (Line 13). Then, the first resource in this group is selected (Line 14), and the completion time of this resource is recorded (Line 15). If the earliest start time of all tasks mapped to a cloud resource is greater than the record time (Line 17), all tasks on this resource are consolidated to the selected resource (Lines 18–20). Otherwise, a cloud resource is reselected and the record time is updated (Lines 22–23).

To illustrate the function *TaskConsolidation()* clearly, Figure 3 gives a visual example based on the schedule in Figure 2. According to the configurations of the five cloud resources in Table 1, they can be divided into two groups, i.e., $G_1 = \{r_1, r_3, r_4\}$ and $G_2 = \{r_2, r_5\}$. The three cloud resources in the first group will be sorted as r_1, r_3, and r_4. The earliest start time of tasks on resource r_3 is not greater than the maximum finish time of tasks on resource r_1, so the task consolidation requirements are not met. On the contrary,

the earliest start time of tasks on resource r_4 is greater than the maximum finish time of task on resource r_1, then the task on r_4 is consolidated to r_1, as illustrated in Figure 3. By comparing Figures 2 and 3, we can observe that resource r_4 can be released, reducing the economic cost of this resource. After task consolidation, according to the charging mode of cloud resources, the charging period of resource r_3 is still 1, and the economic cost remains unchanged. This means that the proposed task consolidation mechanism helps reduce the economic cost without affecting the task execution.

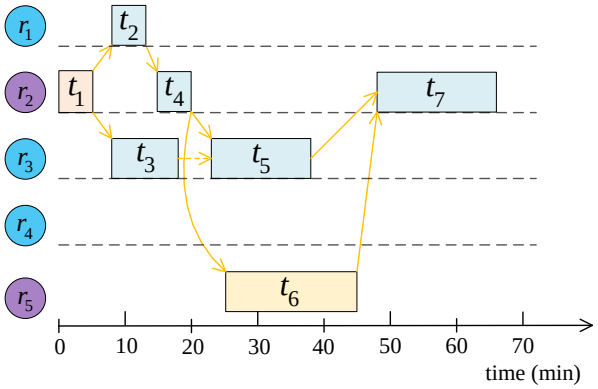


Figure 3. Example of consolidating tasks in different resources.

The start time of a workflow task is constrained by the completion time of collecting the output results from all its predecessors. Besides, if two data-dependent tasks run on the same cloud resource, the data transmission overhead between them is negligible. The above facts motivate us to adjust some tasks to the same resources to simultaneously improve the finish time and economic cost of workflow execution, as illustrated in Algorithm 3.

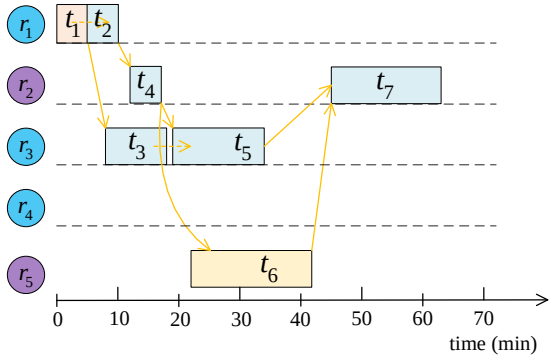
Algorithm 3: Function ***CriticalTaskReschedule()***

Input: Current population P;
Output: A new population P;
1 $CI \leftarrow 0_{1\times n}$;
2 **for** $i = 1 \rightarrow n$ **do**
3 **for** $t_p \in Pred(t_i)$ **do**
4 **if** $x_p == x_i$ **then**
5 Continue;
6 **if** $st(t_i) == ft(t_p) + w(t_p, t_i)/bw$ **then**
7 $CI(i) \leftarrow p$;
8 $Tags \leftarrow 0_{1\times n}$;
9 **for** $i = 1 \rightarrow n$ **do**
10 $p \leftarrow CI(i)$;
11 **if** $p \neq 0 \& Tags(p) == 0 \& Tags(i) == 0$ **then**
12 **if** $rand < 0.5$ **then**
13 $x_p \leftarrow x_i$;
14 $Tags(p) \leftarrow 1$;
15 $Tags(i) \leftarrow 1$;

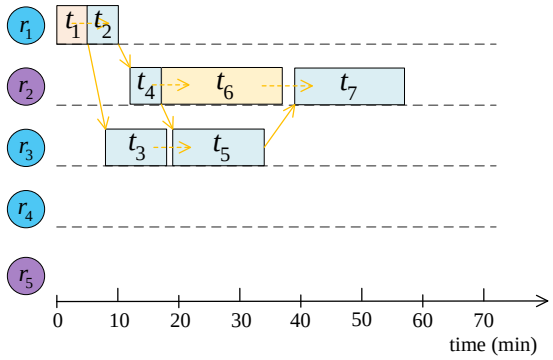
The Function (CriticalTaskReschedule) consists of two parts: identification and adjustment of critical predecessor tasks. For a workflow task, its start time is limited by all its predecessors, and its critical predecessor is defined as the one with the latest result arrival. This function first identifies the critical predecessor for each task (Lines 1–7). The value of the i-th element in CI denotes the index of the critical predecessor for the i-th task.

Due to the complex workflow structure, a task may be the critical predecessor for multiple tasks, and there may be a critical predecessor. To alleviate the offset of critical task adjustments, adjusting the critical predecessor of a task to its mapped resource should satisfy the following three conditions: (1) this task has a critical predecessor, i.e., $p \neq 0$; (2) the critical predecessor has not been adjusted, i.e., $Tags(p) == 0$; (3) this task has not been adjusted as a critical predecessor, i.e., $Tags(i) == 0$.

Take the scheduling scheme in Figure 3 as an example. According to the definition of a critical predecessor in Function (CriticalTaskReschedule), t_1 is the critical predecessor of t_2, and t_6 is the critical predecessor of t_7. Adjusting task t_1 from resource r_2 to r_1, as shown in Figure 4a, the data transmission overhead between task t_1 and t_2 can be eliminated to advance the start time of those tasks including t_2 and its successor tasks. Added to that, adjusting task t_6 from cloud resource r_5 to r_2, as shown in Figure 4b, advances the start/finish time of task t_7. This helps to simultaneously reduce the makespan and economic cost of executing the workflow.



(**a**)



(**b**)

Figure 4. Example of adjusting critical tasks. (**a**) schedule result of adjusting task t_1 from resource r_2 to r_1; (**b**) schedule result of adjusting task t_6 from resource r_5 to r_2.

4. Experimental Studies

The main work of this paper is to propose a task consolidation and a critical task adjustment mechanism, thereby improving the quality of reproducing new populations. The proposal does not involve environmental selection. We employ the proposed two mechanisms to replace the reproduction operators of four classical multi-objective evolutionary algorithms, i.e., NSGA-II [20], HypE [36], MOEA/D [37], and RVEA [38], forming four variants, namely, KEOO-NSGA-II, KEOO-HypE, KEOO-MOEA/D, and KEOO-RVEA.

Then, the four variants are compared with their original versions to verify the effectiveness of the two proposed mechanisms.

4.1. Experimental Setting

We perform the comparison experiments based on five different types of resources provided by the Amazon EC2 cloud platform. These five types of resource instances are t3.nano, t3.micro, t3.small, t3.medium, t3.medium, and t3.large. The main parameters of the five resource types are summarized in Table 4. The length of a charging period is set to 60 seconds, and the bandwidth among resource instances is set to 5.0 Gbps.

Table 4. Parameters for the five types of cloud resources.

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

The five types of workflows from different application domains, i.e., Montage (Astronomy), Epigenomics (Biology), Inspiral (Gravitational physics), CyberShake (Earthquake), and Sipht (Bioinformatics), have been widely used in evaluating cloud workflow scheduling algorithms. For each type of workflow, we select three workflow instances with about 30, 50, and 100 tasks in the experiments. Besides, the DAG diagrams of the workflow instances with around 30 tasks are illustrated in Figure 5. 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 Pegasus repository.

Hypervolume [39] 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 that $\boldsymbol{r} = \{r_1, r_2\}$ 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 P, can be calculated as follows.

$$HV(P) = L(\bigcup_{p \in P}[f_1(p), r_1] \times [f_2(p), r_2]), \tag{11}$$

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

The population size of the eight algorithms is set as 120. The maximum number of fitness evaluations is set as the termination condition for all the five algorithms and is set as $n \times 5 \times 10^3$, where n is the number of decision variables.

The eight algorithms are independently repeated 31 times on each workflow instance to mitigate the impact of random factors. 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.

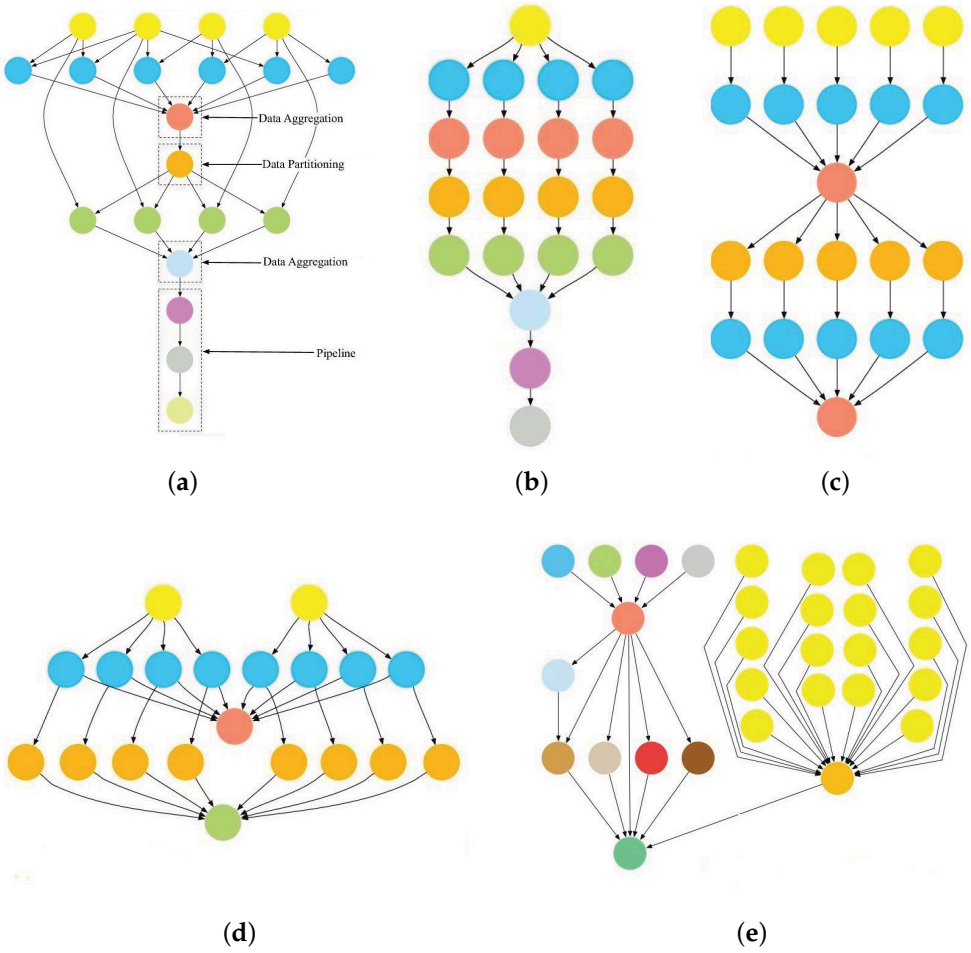


Figure 5. DAG diagrams of workflows with about 30 tasks. (**a**) Montage. (**b**) Epigenomics. (**c**) Inspiral. (**d**) CyberShake. (**e**) Sipht.

4.2. Comparison Results

Table 5 summarizes the average and standard deviation (in brackets) of the hypervolume values for the eight algorithms, i.e., NSGA-II, KEOO-NSGA-II, HypE, KEOO-HypE, MOEA/D, KEOO-MOEA/D, RVEA, and KEOO-RVEA, in scheduling the 15 workflow instances to cloud resources. For each workflow instance, the largest hypervolume value among the eight algorithms is highlighted using a gray background.

Besides, we resort to the Wilcoxon rank-sum test with a significance of 0.1 to differentiate the significant difference between each variant and its source baseline. The signs $-$, $+$, and $\approx$ indicate that the variant is significantly inferior to, superior to, and similar to the corresponding baseline, respectively.

From Table 5, we can observe that except for Inspiral 50 and Cybershake 100, the results marked with a gray background on the other 13 workflow instances are either KEOO-NSGA-II or KEOO-HypE. Specifically, KEOO-NSGA-II significantly improves the hypervolume values of NSGA-II on 10 out of 15 workflow instances, and KEOO-HypE improves its original version on 10 out of 15 workflow instances. The KEOO-NSGA-II and KEOO-HypE are variants of NSGA-II or HypEA by integrating the proposed task consolidation and critical task adjustment mechanisms. The comparison results demonstrate that the proposed mechanisms can effectively improve the performance of the NSGA-II or HypEA.

Table 5. Comparison results for the 8 algorithms on 15 workflows in terms of the hypervolume metric.

Workflows	n	NSGA-II	KEOO-NSGA-II	HypE	KEOO-HypE	MOEA/D	KEOO-MOEA/D	RVEA	KEOO-RVEA
Montage	25	7.013×10^2 (2.23×10^1)	7.113×10^2 + (1.55×10^1)	6.924×10^2 (1.38×10^1)	7.142×10^2 + (8.95×10)	6.078×10^2 (4.61×10^1)	6.253×10^2 + (2.54×10^1)	6.741×10^2 (2.63×10^1)	6.777×10^2 ≈ (2.56×10^1)
	50	1.361×10^3 (3.16×10^1)	1.376×10^3 + (2.99×10^1)	1.370×10^3 (2.92×10^1)	1.392×10^3 + (2.38×10^1)	1.243×10^3 (3.10×10^1)	1.025×10^3 − (4.35×10^2)	1.265×10^3 (3.68×10^1)	1.327×10^3 + (4.59×10^1)
	100	1.353×10^3 (3.88×10^1)	1.476×10^3 + (5.42×10^1)	2.440×10^3 (5.142×10^1)	2.449×10^3 ≈ (7.37×10^1)	2.304×10^3 (5.92×10^1)	1.992×10^3 − (9.18×10^2)	2.211×10^3 (9.12×10^1)	2.449×10^3 + (3.69×10^1)
Epigenomics	24	5.656×10^5 (2.50×10^3)	5.678×10^5 + (2.64×10^3)	5.627×10^5 (2.84×10^3)	5.690×10^5 + (2.17×10^3)	5.201×10^5 (1.13×10^3)	5.173×10^5 − (1.48×10^3)	5.586×10^5 (3.68×10^3)	5.648×10^5 + (2.24×10^3)
	46	1.899×10^6 (5.96×10^3)	1.915×10^6 + (2.04×10^3)	1.893×10^6 (7.10×10^3)	1.909×10^6 + (1.03×10^4)	1.815×10^6 (2.68×10^4)	1.784×10^6 − (3.91×10^4)	1.882×10^6 (9.60×10^3)	1.894×10^6 + (1.21×10^4)
	100	5.056×10^7 (3.56×10^5)	5.066×10^7 ≈ (2.13×10^5)	5.006×10^7 (3.23×10^5)	5.014×10^7 ≈ (3.32×10^5)	4.843×10^7 (5.65×10^5)	4.674×10^7 − (1.06×10^6)	5.022×10^7 (2.78×10^5)	4.962×10^7 − (4.26×10^5)
Inspiral	30	4.078×10^4 (6.49×10^2)	4.093×10^4 ≈ (2.57×10^2)	4.066×10^4 (4.93×10^2)	4.080×10^4 ≈ (1.64×10^2)	3.748×10^4 (5.04×10^2)	3.577×10^4 − (1.70×10^3)	4.016×10^4 (4.21×10^2)	4.025×10^4 ≈ (3.40×10^2)
	50	9.466×10^4 (4.98×10^2)	9.329×10^4 − (8.17×10^2)	9.366×10^4 (1.10×10^3)	9.354×10^4 ≈ (6.47×10^2)	8.945×10^4 (1.33×10^3)	8.360×10^4 − (1.88×10^3)	9.338×10^4 (9.01×10^2)	9.256×10^4 − (1.02×10^3)
	100	1.453×10^5 (1.27×10^3)	1.444×10^5 − (2.01×10^3)	1.426×10^5 (2.17×10^3)	1.457×10^5 + (1.45×10^3)	1.375×10^5 (4.34×10^3)	1.325×10^5 − (3.39×10^3)	1.436×10^5 (1.80×10^3)	1.420×10^5 − (2.35×10^3)
CyberShake	30	2.240×10^5 (8.96×10^3)	2.301×10^5 + (4.07×10^3)	2.235×10^5 (8.31×10^3)	2.485×10^5 + (4.84×10^3)	2.161×10^5 (1.14×10^3)	2.4110×10^5 + (5.98×10^3)	2.1770×10^5 (1.97×10^3)	2.486×10^5 + (4.66×10^3)
	50	3.011×10^5 (2.43×10^3)	3.081×10^5 + (5.98×10^3)	3.038×10^5 (1.77×10^3)	3.065×10^5 + (5.87×10^3)	2.966×10^5 (1.52×10^3)	3.051×10^5 + (4.78×10^3)	3.024×10^5 (2.34×10^3)	3.014×10^5 ≈ (4.23×10^3)
	100	1.971×10^6 (7.59×10^4)	1.930×10^6 − (3.65×10^4)	1.985×10^6 (1.12×10^4)	1.938×10^6 − (2.12×10^4)	1.961×10^6 (7.81×10^4)	1.951×10^6 ≈ (1.45×10^4)	1.968×10^6 (1.26×10^4)	1.944×10^6 − (2.33×10^4)
Sipht	30	8.278×10^4 (3.79×10^2)	8.344×10^4 + (7.04×10^1)	8.302×10^4 (2.74×10^2)	8.342×10^4 + (1.20×10^2)	7.944×10^4 (9.73×10^2)	8.031×10^4 + (2.65×10^2)	8.149×10^4 (4.93×10^2)	8.260×10^4 + (1.15×10^2)
	60	1.991×10^5 (1.17×10^3)	2.032×10^5 + (2.63×10^2)	1.992×10^5 (1.30×10^3)	2.028×10^5 + (4.14×10^3)	1.860×10^5 (4.64×10^3)	1.963×10^5 + (1.12×10^3)	1.938×10^5 (1.64×10^3)	2.009×10^5 + (4.24×10^2)
	100	3.427×10^5 (6.29×10^3)	3.615×10^5 + (8.31×10^2)	3.461×10^5 (4.31×10^3)	3.598×10^5 + (1.98×10^3)	2.996×10^5 (1.19×10^4)	3.502×10^5 + (3.718×10^3)	3.292×10^5 (5.914×10^3)	3.536×10^5 + (1.430×10^3)

MOEA/D and RVEA are representative algorithms of two branches of multi-objective evolutionary algorithms based on decomposition. One branch transforms the multi-objective optimization problem into a set of single-objective subproblems using a set of weight vectors, while the other branch divides the multidimensional objective space into a series of multi-objective subspaces using a set of weight vectors. Compared with NSGA-II or HypEA, these two algorithms and their variants pose poor performance in solving multi-objective optimization problems. The key reason is that the value ranges of the two objectives of the focused problems are far from each other, typically badly-scaled problems. Then, the intersection points between the Pareto-optimal fronts of the focused problems and the weight vectors are unevenly distributed, which weakens the performance of these multi-objective evolutionary algorithms based on decomposition. It can be seen from Table 5 that for more than half of the workflow instances, variant KEOO-RVEA has higher hypervolume values than algorithm RVEA. KEOO-MOEA/D and MOEA/D have similar comparison results. These comparison results demonstrate that the proposed task consolidation and critical task adjustment mechanisms can effectively improve the performance of multi-objective evolutionary algorithms based on decomposition in solving multi-objective cloud workflow scheduling problems.

From Table 5, the difference in hypervolume values obtained by the eight algorithms is not apparent on the same workflow instances. This is because the reference point for calculating the hypervolume value is set based on the nadir point of the initial population, which is often far from the output populations of these algorithms. Thus, each algorithm can obtain a very large hypervolume value, resulting in slight differences.

It can be seen from Figure 5 that among the five types of workflows, the Montage workflow has the most complex structure. From Table 5, we also observe that on the four workflow instances derived from this application, the proposed task consolidation and critical task adjustment mechanisms can greatly improve the hypervolume values of the four classical multi-objective evolutionary algorithms. These improvements demonstrate the effectiveness of the proposed mechanisms in dealing with complex workflow structures.

To intuitively compare the convergence and diversity of the eight multi-objective workflow scheduling algorithms, Figure 6 illustrates the distribution of their output populations on workflow instances Inspiral, Epigenomics, Montage, CyberShake, and Sipht with around 30 tasks as well as Sipht with around 100 tasks.

In Figure 6, the populations obtained by the algorithms using the proposed KEOO are shown in red, and different algorithms are distinguished by different icons. The first impression of the six sub-figures is that the red icons are distributed at the forefront. This indicates that the proposed KEOO improves the existing multi-objective evolutionary algorithms on both makespan and economic cost. These comparison results are consistent with those in Table 5. The population distribution of the variant with the proposed KEOO is basically consistent with that of the original algorithm. This is because they all use the same environmental selection operator. The advantage of the four variants is that they employ the task consolidation mechanism to reduce economic cost, and the critical task adjustment mechanism to reduce makespan and economic cost simultaneously.

On workflow instances Inspiral 30 and Epigenomics 24, although the solutions of the algorithms embedded with the proposed mechanisms distribute at the forefront, they are similar to the solutions of their corresponding original algorithms. The reason can be attributed to the fact that these two workflow instances can be divided into multiple independent simple pipelines, and the corresponding optimization problems are relatively easy. The traditional multi-objective evolutionary algorithms can solve these simple problems well, leaving limited room for improvement.

On workflow instances Montage 25 and Cybershake 30, solutions obtained by KEOO-NSGA-II and KEOO-HypE are obviously separated from the solutions of their original algorithms. The reason is that these two workflow instances are relatively complex. When scheduling such workflow instances, the start and completion time of different resources vary greatly, leaving room for the task consolidation mechanism to play the advantage of reducing economic cost. A distinctive feature of workflow instance Cybershake 30 is that one of its tasks has a large number of predecessors, and these predecessors no longer have predecessors. Besides, the tasks in Cybershake 30 need to transfer a large amount of data, resulting in many large idle time gaps between tasks. The proposed critical task adjustment mechanism can adjust some critical tasks to idle time gaps to simultaneously reduce the makespan and economic cost.

By comparing Figure 6e,f, we can see that the advantage of algorithms embedded with the proposed mechanisms on Sipht 100 is much more obvious than on Sipht 30. This comparison result illustrates that the proposed mechanisms help accelerate population convergence and effectively deal with the explosively growing search space.

To intuitively demonstrate the advantages of the proposed mechanisms in accelerating the convergence of multi-objective evolutionary algorithms, Figure 7 illustrates the changes in the hypervolume values of eight algorithms with the evolution process. We can observe that the hypervolume values of all eight algorithms rise rapidly in the early stage of the evolutionary search. The reason is that the cloud resource pool is large enough, meaning a wide range of decision variable values. The quality of random initial solutions is poor, and it is relatively easy for evolutionary operators to push them toward the Pareto-optimal fronts. Concerning the hypervolume metric, the growth rate of the algorithms embedded with the proposed mechanisms is faster than that of their original algorithms throughout the search process. These results demonstrate that the proposed mechanisms are capable of facilitating the convergence rate.

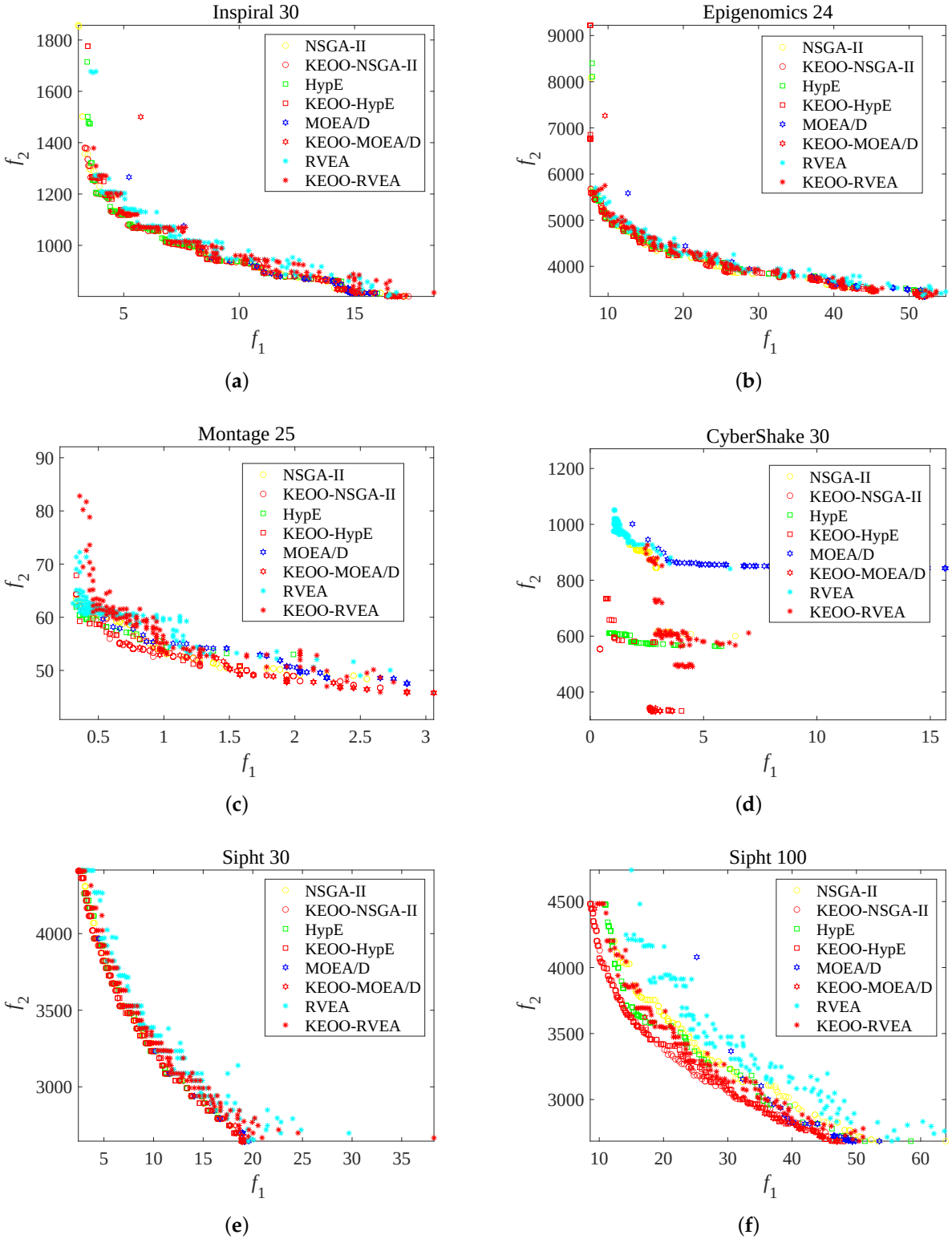


Figure 6. Distributions of populations obtained by the 8 algorithms on Inspiral, Epigenomics, Montage, CyberShake, and Sipht with around 30 tasks as well as Sipht with around 100 tasks. (**a**) on Inspiral with 30 tasks; (**b**) on Epigenomics with 24 tasks; (**c**) on Montage with 25 tasks; (**d**) on CyberShake with 30 tasks; (**e**) on Sipht with 30 tasks; (**f**) on Sipht with 100 tasks.

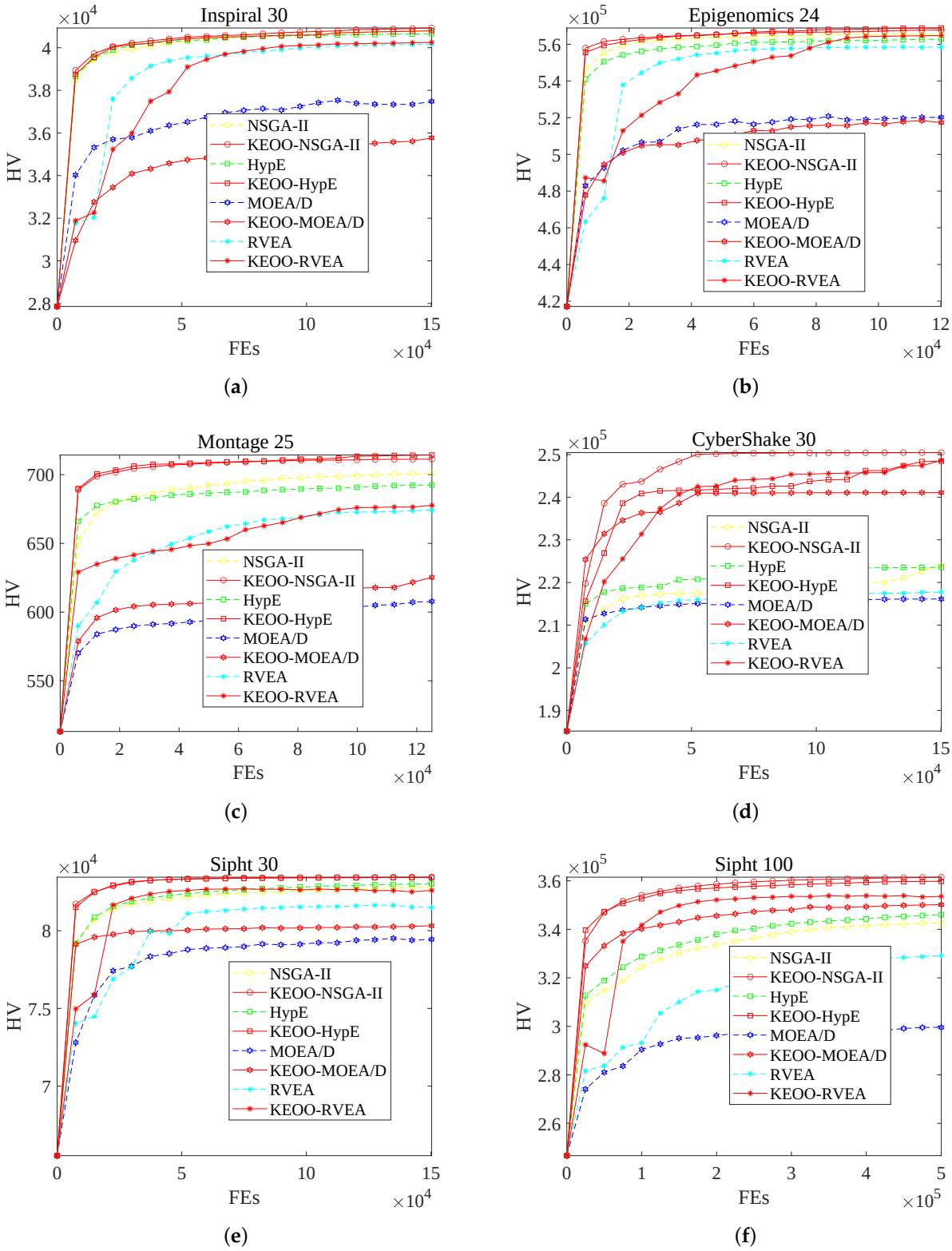


Figure 7. Change of hypervolume values with the advance of evolution. (**a**) on Inspiral with 30 tasks; (**b**) on Epigenomics with 24 tasks; (**c**) on Montage with 25 tasks; (**d**) on CyberShake with 30 tasks; (**e**) on Sipht with 30 tasks; (**f**) on Sipht with 100 tasks.

For the execution time, the comparison results of the eight algorithms are shown in Table 6. Obviously, the execution time of the variants embedded in the proposed mechanism in this paper is greater than that of the original algorithms. The increased execution cost

mainly comes from the proposed mechanism. Fortunately, the execution time of the four variants is about the same order of magnitude as that of the original algorithms, and these variants can obtain populations with better convergence and diversity.

Table 6. Execution time (second) of the eight algorithms.

Workflows	n	NSGA-II	KEOO-NSGA-II	HypE	KEOO-HypE	MOEA/D	KEOO-MOEA/D	RVEA	KEOO-RVEA
CyberShake	30	1.275×10^1 (1.03×10^0)	2.333×10^1 (3.95×10^{-1})	1.568×10^1 (9.12×10^{-1})	2.391×10^1 (5.45×10^{-1})	1.113×10^1 (1.02×10^0)	2.381×10^1 (2.61×10^{-1})	1.734×10^1 (1.04×10^0)	2.814×10^1 (6.76×10^{-1})
	50	3.719×10^1 (1.33×10^0)	6.274×10^1 (8.70×10^{-1})	3.846×10^1 (1.64×10^0)	6.453×10^1 (7.01×10^{-1})	3.810×10^1 (1.40×10^0)	6.390×10^1 (1.01×10^0)	4.077×10^1 (1.20×10^0)	7.199×10^1 (1.76×10^0)
	100	1.433×10^2 (6.95×10^0)	2.453×10^2 (3.42×10^0)	1.695×10^2 (9.06×10^0)	2.505×10^2 (4.02×10^0)	1.240×10^2 (2.51×10^0)	2.479×10^2 (2.63×10^0)	1.464×10^2 (7.05×10^0)	2.678×10^2 (5.68×10^0)

5. Conclusions

This paper mathematically formulates the workflow scheduling problem in cloud computing as a bi-objective optimization problem. Then, it explores the knowledge of cloud resources and workflow structures to design a task consolidation mechanism to reduce economic cost by reducing the number of cloud resources; and a critical task-based search operator to selectively move the critical precursors of some tasks to the same resources to eliminate the data transmission overhead between them, striving to improve the economic cost and completion time simultaneously. Based on real-world workflows and cloud platforms, the proposed mechanisms are embedded into four classical multi-objective evolutionary algorithms to verify their effectiveness in improving population convergence and diversity. One disadvantage of the proposed mechanisms is that they bring extra time overhead to the classical algorithms.

Cloud workflow scheduling is a representative grey-box problem, and it is interesting to further mine the knowledge of the workflows and cloud resources to derive efficient scheduling algorithms. Through experimental analysis, we can see that the time overhead of evolutionary optimization can not be ignored. Therefore, another potential direction is to design an effective parallel evolutionary framework to shorten the time overhead to support cloud workflow scheduling in real-time and uncertain situations [40].

Author Contributions: Conceptualization, L.X. and J.L.; methodology, J.L.; software, L.X. and R.W.; validation, L.X., R.W., J.C. and J.L.; formal analysis, R.W.; investigation, R.W.; resources, J.C.; data curation, J.C.; writing—original draft preparation, L.X.; writing—review and editing, R.W., J.C. and J.L.; visualization, J.C.; supervision, J.L.; project administration, J.L.; funding acquisition, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research work is supported by the National Natural Science Foundation of China (61773120), the Special Projects in Key Fields of Universities in Guangdong (2021ZDZX1019).

Data Availability Statement: All data used during the study appear in the submitted article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, H.; Wen, J.; Pedrycz, W.; Wu, G. Big data processing workflows oriented real-time scheduling algorithm using task-duplication in geo-distributed clouds. *IEEE Trans. Big Data* **2018**, *6*, 131–144. [CrossRef]
2. Bugingo, E.; Zhang, D.; Chen, Z.; Zheng, W. Towards decomposition based multi-objective workflow scheduling for big data processing in clouds. *Clust. Comput.* **2021**, *24*, 115–139. [CrossRef]
3. Molnár, B.; Benczúr, A. The application of directed hyper-graphs for analysis of models of information systems. *Mathematics* **2022**, *10*, 759. [CrossRef]
4. Cong, P.; Li, L.; Zhou, J.; Cao, K.; Wei, T.; Chen, M.; Hu, S. Developing user perceived value based pricing models for cloud markets. *IEEE Trans. Parallel Distrib. Syst.* **2018**, *29*, 2742–2756. [CrossRef]
5. Jung, A.; Gsell, M.A.; Augustin, C.M.; Plank, G. An integrated workflow for building digital twins of cardiac electromechanics—A multi-fidelity approach for personalising active mechanics. *Mathematics* **2022**, *10*, 823. [CrossRef]

6. Farid, M.; Latip, R.; Hussin, M.; Hamid, N.A.W.A. Scheduling scientific workflow using multi-objective algorithm with fuzzy resource utilization in multi-cloud environment. *IEEE Access* **2020**, *8*, 24309–24322. [CrossRef]
7. Masdari, M.; ValiKardan, S.; Shahi, Z.; Azar, S.I. Towards workflow scheduling in cloud computing: A comprehensive analysis. *J. Netw. Comput. Appl.* **2016**, *66*, 64–82. [CrossRef]
8. Zhang, M.; Li, H.; Liu, L.; Buyya, R. An adaptive multi-objective evolutionary algorithm for constrained workflow scheduling in Clouds. *Distrib. Parallel Databases* **2018**, *36*, 339–368. [CrossRef]
9. Zhu, Z.; Zhang, G.; Li, M.; Liu, X. Evolutionary multi-objective workflow scheduling in cloud. *IEEE Trans. Parallel Distrib. Syst.* **2016**, *27*, 1344–1357. [CrossRef]
10. Hosseinzadeh, M.; Ghafour, M.Y.; Hama, H.K.; Vo, B.; Khoshnevis, A. Multi-objective task and workflow scheduling approaches in cloud computing: A comprehensive review. *J. Grid Comput.* **2020**, *18*, 327–356. [CrossRef]
11. Zhou, X.; Zhang, G.; Sun, J.; Zhou, J.; Wei, T.; Hu, S. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Gener. Comput. Syst.* **2019**, *93*, 278–289. [CrossRef]
12. Kumar, M.S.; Tomar, A.; Jana, P.K. Multi-objective workflow scheduling scheme: A multi-criteria decision making approach. *J. Ambient. Intell. Humaniz. Comput.* **2021**, *12*, 10789–10808. [CrossRef]
13. Ye, X.; Liu, S.; Yin, Y.; Jin, Y. User-oriented many-objective cloud workflow scheduling based on an improved knee point driven evolutionary algorithm. *Knowl. Based Syst.* **2017**, *135*, 113–124. [CrossRef]
14. Pham, T.P.; Fahringer, T. Evolutionary multi-objective workflow scheduling for volatile resources in the cloud. *IEEE Trans. Cloud Comput.* **2022**, *10*, 1780–1791. [CrossRef]
15. Rodriguez, M.A.; Buyya, R. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4041. [CrossRef]
16. Zhan, Z.H.; Liu, X.F.; Gong, Y.J.; Zhang, J.; Chung, H.S.H.; Li, Y. Cloud computing resource scheduling and a survey of its evolutionary approaches. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–33. [CrossRef]
17. Durillo, J.J.; Nae, V.; Prodan, R. Multi-objective energy-efficient workflow scheduling using list-based heuristics. *Future Gener. Comput. Syst.* **2014**, *36*, 221–236. [CrossRef]
18. Fard, H.M.; Prodan, R.; Fahringer, T. Multi-objective list scheduling of workflow applications in distributed computing infrastructures. *J. Parallel Distrib. Comput.* **2014**, *74*, 2152–2165. [CrossRef]
19. Han, P.; Du, C.; Chen, J.; Ling, F.; Du, X. Cost and makespan scheduling of workflows in clouds using list multiobjective optimization technique. *J. Syst. Archit.* **2021**, *112*, 101837. [CrossRef]
20. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
21. Wu, Q.; Zhou, M.; Zhu, Q.; Xia, Y.; Wen, J. MOELS: Multiobjective evolutionary list scheduling for cloud workflows. *IEEE Trans. Autom. Sci. Eng.* **2020**, *17*, 166–176. [CrossRef]
22. 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.* **2019**, *49*, 2912–2926. [CrossRef] [PubMed]
23. Gupta, R.; Gajera, V.; Jana, P.K. An effective multi-objective workflow scheduling in cloud computing: A PSO based approach. In Proceedings of the 2016 Ninth International Conference on Contemporary Computing, Noida, India, 11–13 August 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1–6.
24. Wang, Y.; Zuo, X. An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1079–1094. [CrossRef]
25. Ismayilov, G.; Topcuoglu, H.R. Neural network based multi-objective evolutionary algorithm for dynamic workflow scheduling in cloud computing. *Future Gener. Comput. Syst.* **2020**, *102*, 307–322. [CrossRef]
26. Hussain, M.; Wei, L.F.; Abbas, F.; Rehman, A.; Ali, M.; Lakhan, A. A multi-objective quantum-inspired genetic algorithm for workflow healthcare application scheduling with hard and soft deadline constraints in hybrid clouds. *Appl. Soft Comput.* **2022**, *128*, 109440. [CrossRef]
27. Abbasian-Naghneh, S.; Kalbasi, R. Implementation of ANN and GA on building with PCM at various setpoints, PCM types, and installation locations to boost energy saving and CO_2 saving. *Eng. Anal. Bound. Elem.* **2022**, *144*, 110–126. [CrossRef]
28. Abed-Alguni, B.H.; Alawad, N.A. Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments. *Appl. Soft Comput.* **2021**, *102*, 107113. [CrossRef]
29. 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.* **2018**, *83*, 14–26. [CrossRef]
30. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F.S. Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimization algorithm. *J. Netw. Syst. Manag.* **2021**, *29*, 1–34. [CrossRef]
31. De Maio, V.; Kimovski, D. Multi-objective scheduling of extreme data scientific workflows in Fog. *Future Gener. Comput. Syst.* **2020**, *106*, 171–184. [CrossRef]
32. Calheiros, R.N.; Buyya, R. Meeting deadlines of scientific workflows in public clouds with tasks replication. *IEEE Trans. Parallel Distrib. Syst.* **2014**, *25*, 1787–1796. [CrossRef]
33. Coello, C.A.C.; Brambila, S.G.; Gamboa, J.F.; Tapia, M.G.C.; Gómez, R.H. Evolutionary multiobjective optimization: Open research areas and some challenges lying ahead. *Complex Intell. Syst.* **2020**, *6*, 221–236. [CrossRef]

34. Chen, H.; Cheng, R.; Wen, J.; Li, H.; Weng, J. Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations. *Inf. Sci.* **2020**, *509*, 457–469. [CrossRef]
35. Li, B.; Li, J.; Tang, K.; Yao, X. Many-objective evolutionary algorithms: A survey. *ACM Comput. Surv. (CSUR)* **2015**, *48*, 1–35. [CrossRef]
36. Bader, J.; Zitzler, E. HypE: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.* **2011**, *19*, 45–76. [CrossRef]
37. Zhang, Q.; Li, H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
38. Cheng, R.; Jin, Y.; Olhofer, M.; Sendhoff, B. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **2016**, *20*, 773–791. [CrossRef]
39. Zitzler, E.; Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Trans. Evol. Comput.* **1999**, *3*, 257–271. [CrossRef]
40. Chen, H.; Zhu, X.; Liu, G.; Pedrycz, W. Uncertainty-aware online scheduling for real-time workflows in cloud service environment. *IEEE Trans. Serv. Comput.* **2021**, *14*, 1167–1178. [CrossRef]