mathematics

MDPI

Article

Evolutionary Optimization of Energy Consumption and Makespan of Workflow Execution in Clouds

Lining Xing [1], Jun Li [2,*], Zhaoquan Cai [3,4] and Feng Hou [5]

[1] School of Mathematics and Big Data, Foshan University, Foshan 528225, China; xinglining@nudt.edu.cn
[2] School of Management, Hunan Institute of Engineering, Xiangtan 411104, China
[3] Shanwei Institute of Technology, Shanwei 516600, China; cai@hzu.edu.cn
[4] School of Computer Science and Engineering, Huizhou University, Huizhou 516007, China
[5] School of Mathematical and Computational Sciences, Massey University, Palmerston North 4442, New Zealand; f.hou@massey.ac.nz
* Correspondence: jli@hnie.edu.cn

Abstract: Making sound trade-offs between the energy consumption and the makespan of workflow execution in cloud platforms remains a significant but challenging issue. So far, some works balance workflows' energy consumption and makespan by adopting multi-objective evolutionary algorithms, but they often regard this as a black-box problem, resulting in the low efficiency of the evolutionary search. To compensate for the shortcomings of existing works, this paper mathematically formulates the cloud workflow scheduling for an infrastructure-as-a-service (IaaS) platform as a multi-objective optimization problem. Then, this paper tailors a knowledge-driven energy- and makespan-aware workflow scheduling algorithm, namely EMWSA. Specifically, a critical task adjustment-based local search strategy is proposed to intelligently adjust some critical tasks to the same resource of their successor tasks, striving to simultaneously reduce workflows' energy consumption and makespan. Further, an idle gap reuse strategy is proposed to search the optimal energy consumption of each non-critical task without affecting the operation of other tasks, so as to further reduce energy consumption. Finally, in the context of real-world workflows and cloud platforms, we carry out comparative experiments to verify the superiority of the proposed EMWSA by significantly outperforming 4 representative baselines on 19 out of 20 workflow instances.

Keywords: mathematical model; cloud computing; workflow scheduling; evolutionary algorithm; multi-objective optimization

MSC: 97M40

Citation: Xing, L.; Li, J.; Cai, Z.; Hou, F. Evolutionary Optimization of Energy Consumption and Makespan of Workflow Execution in Clouds. *Mathematics* **2023**, *11*, 2126. https://doi.org/10.3390/math11092126

Academic Editor: José Antonio Sanz

Received: 13 March 2023
Revised: 19 April 2023
Accepted: 24 April 2023
Published: 30 April 2023

1. Introduction

Cloud computing is a revolutionary paradigm which enables the on-demand delivery of resources and services over the Internet [1]. This allows customers to access a wide range of computing services on a pay-per-use basis without investing in additional hardware and software infrastructures. Relying on the advantages of economy of scale, high scalability, flexibility, fault-tolerant, and lower costs, cloud computing is attracting applications from enterprises and governments. Meanwhile, cloud computing has proliferated rapidly over the past decade [2–4].

To process the ever-growing big data in various fields [5–7], cloud computing providers are building more and more hyper-scale cloud data centers around the world. A data center often deploys millions of high-performance servers, network facilities, and storage devices [8]. As a consequence, the massive facilities in cloud data centers consume enormous amounts of electric energy. It is reported that cloud data centers around the world have consumed nearly 2% of worldwide electricity in 2020, and that this will increase to 8% by 2030 [9]. Such high energy consumption unavoidably causes a large amount of carbon

dioxide emissions, which further give rise to environmental deterioration issues [10–12]. Furthermore, high energy consumption leads to high operating costs for cloud platforms. The electric energy consumed by Amazon's cloud data centers cost nearly 20% of its total budget. Thus, reducing the energy consumption of cloud data centers not only is crucial to the implementation of sustainable computing but also lowers the monetary cost and, thus, improves the market competitiveness for the cloud providers [13,14].

Workflow has been a popular paradigm that supports the complicated process of big data applications on cloud platforms [15,16]. A workflow often contains a set of tasks and the data dependencies among tasks can be modeled as a Directed Acyclic Graph (DAG). In general, the processing of workflows is computing- and data-intensive with a large amount of data being produced and transferred. Taking satellite observation image processing as an example, it involves geometric rectification, data filtering, object classification, change detection, and other phases. Furthermore, a large amount of data needs to be transferred among tasks belonging to different phases [17]. It is worth noting that most workflow applications have to output results as fast as possible.

Confronting such scenarios, workflow scheduling in cloud computing should be formulated as a multi-objective optimization problem aimed at optimizing two conflicting objectives: energy consumption and makespan. In recent years, many works have suggested heuristics [18], meta-heuristics [19,20], and artificial neural networks [21–23] to solve this problem. However, most existing multi-objective workflow scheduling approaches for cloud platforms fail to capture the inherent characteristics of cloud resources and workflows. From the perspective of cloud resource characteristics, the dynamic voltage/frequency scaling (DVFS) technique, which enables dynamic adjusting of voltages and frequencies of processors [24], is commonly used for energy conservation in high-performance computing systems. By dynamically adjusting processors' voltages and frequencies, different trade-offs between energy consumption and performance can be obtained. However, the DVFS technique has not yet been fully explored in multi-objective evolutionary algorithms to balance energy consumption and makespan. From the perspective of workflow characteristics, the data transmission time between tasks executed on the same cloud resource can be negligible. Then, by adjusting the critical predecessor of a task to the same resource it is capable of eliminating the data transmission time, which is promising to simultaneously reduce the energy consumption and makespan. However, the above characteristic are rarely explored to improve the multi-objective workflow scheduling algorithms.

Considering the above facts, we formulate workflow scheduling in cloud computing as a multi-objective optimization problem aimed at optimizing two conflicting objectives: energy consumption and makespan. To solve the problem, we design an efficient multi-objective workflow algorithm by exploring the characteristics of cloud resources and workflows, especially the dynamic voltage/frequency scaling technique and workflow structure. The proposal embraces two new strategies. The first one is a critical task adjustment-based local search strategy, which intelligently adjusts some critical tasks to the same resource as their successor tasks, striving to simultaneously reduce workflows' energy consumption and makespan. The second one is an idle gap reuse strategy, which searches the optimal energy consumption of each non-critical task without affecting the operation of other tasks, so as to further reduce energy consumption. At last, we verify the superiority of the proposal by comparing it with four baselines in the context of real-world workflows and cloud platforms.

The rest of this paper is organized as follows. Section 3 provides the models for workflows and cloud resources, and then formulates the multi-objective optimization problem. Section 4 develops the algorithms. Section 5 provides the performance evaluation. Section 6 concludes this paper and discusses two future research directions.

2. Related Work

Over the past decade, simultaneously optimizing the energy consumption and makespan of cloud workflows has attracted a lot of attention, and numerous relevant methods have been reported [2,25]. They can be roughly partitioned into two branches: heuristics-based and meta-heuristics-based workflow scheduling algorithms.

The list-based workflow scheduling methods, represented by the heterogeneous earliest first time (HEFT) and its variants, are famous heuristics. They have been embedded into multi-objective frameworks to balance multiple conflicting objectives in scheduling cloud workflows. For instance, Durillo et al. [18] extended the heterogeneous earliest finish time algorithm to make trade-offs between energy consumption and makespan. Faragardi et al. [26] suggested a greedy resource provisioning and list-based scheduling method to optimize cost and makespan while meeting the budget constraints. Medara et al. [26] introduced a list-based energy-efficient workflow algorithm to optimize energy efficiency, execution cost, and resource utilization at the same time. Although these approaches based on specific heuristic strategies can always output feasible schedules, they only search for part of the solution space, leading to the oversight of some promising solutions.

For example, Li et al. [15] proposed five energy- and cost-aware scheduling strategies to reduce the energy consumption and cost of workflow execution. Pan et al. [27] developed a strength Pareto-based multi-objective clustering evolutionary algorithm to minimize the cost and energy consumption of multiple workflows with deadlines in mobile edge computing. Mohammadzadeh et al. [28] combined the antlion optimization algorithm and the grasshopper optimization algorithm to balance the makespan, energy consumption, execution cost, and throughput. Mohammadzadeh et al. [29] combined the antlion optimizer with a sine cosine algorithm to solve the workflow scheduling problem considering four conflicting objectives: makespan, cost, energy consumption, and throughput. Hussain et al. [30] designed new genetic operators by referring to the principles of quantum mechanics and quantum rotation gate to simultaneously optimize makespan and energy consumption. Paknejad et al. [31] combined the chaotic systems into the population initialization, crossover/mutation operators of the preference-based, multi-objective coevolutionary framework to optimize makespan, execution cost, and energy consumption. Based on the classical multi-objective evolutionary algorithm NSGA-II, Peng et al. [32] developed a multi-objective scheduling approach to balance the cost and energy consumption of workflows. Ismayilov et al. [21] incorporated an artificial neural network with the NSGA-II algorithm to balance makespan, cost, energy, and degree of imbalance, reliability, and utilization. Tarafdar et al. [33] suggested two energy and makespan-aware approaches, including a linear weighted sum strategy and an ant colony optimization policy, to optimize energy consumption and the makespan of workflow execution. To pursue a sound trade-off between the makespan and the energy consumption of workflow execution, Xia et al. [34] developed an initialization scheduling sequence strategy and a longest common sub-sequence preservation strategy to improve a multi-objective genetic algorithm. However, most existing multi-objective workflow scheduling approaches for cloud platforms cannot exploit the inherent characteristics of cloud resources and workflows.

3. Mathematical Models

This section models the workflows and cloud resources, and then presents the mathematical formulation for the multi-objective workflow scheduling problem in cloud computing. For the convenience of reference, we summarize the main notations in Table 1.

Table 1. Notations used in the study.

Notation	Definition
V	set of tasks in the workflow
v_i	i-th task in the workflow
$P(v_i)$	set of task v_i's direct precursors
$S(v_i)$	set of task v_i's direct successors
B_i	all of the tasks being executed before v_i on the same resource
V_k	all of the tasks being mapped to resource r_k^τ
r_k^τ	k-th resource instance with type τ
$\tau \in \{1, 2, \cdots, m\}$	τ-th resource type
E	set of directed edges among tasks
$w(e_{p,i})$	size of data being transferred from task v_p to task v_i
$st_{v_i,k}$	start time of task v_i on resource r_k^τ
$ft_{v_i,k}$	finish time of task v_i on resource r_k^τ
$et_{v_i,k}$	execution time of task v_i on resource r_k^τ
$p(\tau, f(t))$	power consumption of a resource with type τ

3.1. Workflow and Resource Model

To facilitate the big data processing workflows to make the best use of the cloud resources, researchers and engineers widely model them as directed acyclic graphs (DAGs). A workflow corresponds to a unique directed acyclic graph, denoted as $G = \{V, E\}$, where $V = \{v_1, v_2, \cdots, v_n\}$ denotes n tasks in the workflow and $E \subseteq V \times V$ is the set of edges denoting precedence-constraints between tasks. An edge $e_{i,j} \in E$ means that the v_j cannot be executed before receiving v_i's output data. Then, task v_i is defined as a direct precursor of task v_j, and v_j is defined as a direct successor of v_i. For any task v_i, all of its direct precursors are denoted as set $P(v_i)$, and all of its direct successors are denoted as set $S(v_i)$.

Figure 1 gives a visual example of a workflow. Its DAG model can be described as $G = \{V, E\}$, where $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{e_{1,2}, e_{1,3}, e_{2,4}, e_{3,5}, e_{4,5}\}$. The edge $e_{2,4}$ portrays the precedence constraint from v_1 to v_4, meaning that the start of task v_4 is constrained by v_2's output data. From Figure 1, we can also see that the set of v_5's direct precursors is $P(v_5) = \{v_3, v_4\}$ and the set of v_1's direct successors is $S(v_1) = \{v_2, v_3\}$.

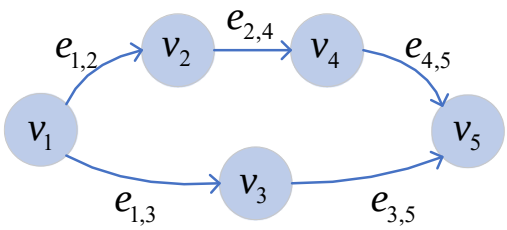


Figure 1. An example of a workflow with five tasks.

Similar to other works [21,35,36], this paper also focuses on the IaaS paradigm, where cloud service providers generally offer various types of cloud resources at different configurations, such as CPU frequency, energy power, network bandwidth, and memory size. All types of cloud resources can be summarized as $\Gamma = \{1, 2, \cdots, m\}$, where m denotes the number of resource types and $\tau \in \Gamma$ denotes the τ-th resource type. Then, a resource instance of type τ in cloud platforms can be described as $r_k^\tau = \{k, con(\tau), p(\tau, t)\}$, in which k and $con(\tau)$ represent its index and configurations, and $p(\tau, f(t))$ denotes its power consumption with CPU frequency f at time instance t.

For a DVFS-enabled cloud resource, its power consumption $p(\tau, f(t))$ can be described as follows [13]:

$$p(\tau, f(t)) = p_\tau^s + \alpha_\tau \cdot v(t)^2 \cdot f(t), \quad (1)$$

where p_τ^s denotes the static power consumption, i.e., the power consumption when the instance is completely idle; $\alpha_\tau \cdot v(t)^2 \cdot f(t)$ denotes the dynamic power consumption caused by processing workloads. Furthermore, α_τ denotes the proportionality coefficient for the resource type τ; $v(t)$ and $f(t)$ denote the supply voltage and frequency at time t.

Since the frequency and supply voltage are approximately linear, Equation (1) can be simplified as:

$$p(\tau, f(t)) = p_\tau^s + \alpha'_\tau \cdot f(t)^3. \tag{2}$$

3.2. Problem Formulation

From the perspective of the end-users, the available resources in cloud platforms are infinite. In this paper, we build a resource pool considering the maximum resource demands of the workflow. Assuming the maximum parallelism of a workflow is p and there are m types of resources, we formally describe the resource pool as follows:

$$R = \{r_1^1, r_2^1, \cdots, r_p^1, r_{p+1}^2, r_{p+1}^2, \cdots, r_{2\cdot p}^2, \cdots, r_{m\cdot p}^m\}. \tag{3}$$

Workflow scheduling in DVFS-enabled cloud platforms involves three types of decision vectors: task sequencing, task runtime, and mappings from tasks to resources. To simplify the optimization process, we sort the workflow tasks based on their downward rank [37] and employ their minimum runtime when evolving the mappings from tasks to resources. The decision vector $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}$ represents the mappings from tasks to resources, in which the value of the i-th decision variable x_i denotes the index of the resource mapped to the i-th task. It is intuitive that the value of each decision variable is selected from the set $\{1, 2, \cdots, m \cdot p\}$.

With a decision vector, the mapped resource of task v_i is assumed to be r_k^τ, on which the tasks before v_i can be described as the following set:

$$B_i = \{v_p | O(v_p) < O(v_i)\}, \tag{4}$$

where $O(v_p)$ represents the order number of task v_p on resource r_k^τ.

The start time $st(v_i, k)$ of task v_i on resource r_k^τ can be obtained as follows:

$$st(v_i, k) = \max\{\max_{v_b \in B_i} ft(v_b, k), \max_{v_p \in P(v_i)} \{ft(v_p, *) + dt(v_p, v_i)\}\}, \tag{5}$$

where $ft(v_b, k)$ represents v_b's finish time on resource r_k^τ, $ft(v_p, *)$ represents v_p's finish time on its mapped resource, and $dt(v_p, v_i)$ represents the data transfer duration from v_p to v_i.

We use the symbol $et(v_i, k)$ to represent the minimum execution time of task v_i on the mapped resource r_k^τ. The relationships among $st(v_i, k)$, $et(v_i, k)$, and $ft(v_i, k)$ can be summarized as follows:

$$ft(v_i, k) = st(v_i, k) + et(v_i, k). \tag{6}$$

Due to the precedence constraints between tasks, a task can only start running after receiving the output results of all of the direct precursors, which leads to the following constraint:

$$st(v_i, k) \geq \max_{v_p \in P(v_i)} \{ft(v_p, r(v_i)) + \frac{I\{r(v_i) \neq r(v_p)\} \times w(e_{p,i})}{bw}\}, \forall v_i \in V, \tag{7}$$

where $I\{\cdot\}$ is an indicator function, when v_p and v_i are mapped to the same resource, $I\{\cdot\}$ is 0; otherwise, it is 1. The indicator function reflects the fact that when two dependent tasks are processed by the same resource, their data transfer time can be negligible and assumed to be zero. bw represents the bandwidth.

With a decision vector, all of the tasks being mapped to resource r_k^τ can be denoted with the following set:

$$V_k = \{v_i | x_i = k, i \in \{1, 2, \cdots, n\}\}. \tag{8}$$

Then, the power-up time t_u and off time t_o of resource r_k^τ are as follows:

$$\begin{aligned} t_u &= \min_{v_i \in V_k} \{st(v_i, k) - \max_{v_p \in P(v_i)} dt(v_p, v_i)\}, \\ t_o &= \max_{v_i \in V_k} \{ft(v_i, k) + \max_{v_s \in S(v_i)} dt(v_i, v_s)\}. \end{aligned} \tag{9}$$

Based on the above analysis, we formulate the first optimization objective to minimize energy consumption as follows:

$$\text{Minimize } f_1(\mathbf{x}) = \sum_{k=1}^{m \cdot p} \int_{t_u}^{t_o} p_\tau^s + \alpha_\tau' \cdot f_k(t)^3 dt. \tag{10}$$

The second optimization objective is to minimize the workflow's makespan, which refers to the maximum finish time of all of the tasks by considering both the task execution time and the data transfer time among the tasks. This optimization objective is formulated as follows:

$$\text{Minimize } f_2(\mathbf{x}) = \max_{v_i \in V} ft(v_i, *). \tag{11}$$

In sum, the model of multi-objective workflow scheduling in cloud platforms is summarized as follows:

$$\begin{cases} \text{Minimize} & \boldsymbol{f}(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x})], \\ \text{S.t.} & \\ & \boldsymbol{x} \in \{1, 2, \cdots, m \cdot p\}^n, \\ & (7). \end{cases} \tag{12}$$

From (12), we derive that the focused optimization problem is challenging. Its decision variables are discrete, and their relationships are diverse and complex. Furthermore, its objective functions include nonlinear expressions. These characteristics make this problem intractable. Thus, we strive to design a knowledge-based optimization algorithm to handle it.

The focused problem is a representative multi-objective optimization problem (MOP). A key feature of a MOP is that there is no single solution that is optimal in terms of all objectives. Instead, there exists a set of compromise solutions, which are called the Pareto-optimal set in the decision space and the Pareto-optimal front in the objective space [38,39].

Pareto-Dominance: Regarding two feasible solutions $\mathbf{x}_1$ and $\mathbf{x}_2$, $\mathbf{x}_1$ is defined to Pareto-dominate $\mathbf{x}_2$ (denoted by $\mathbf{x}_1 \prec \mathbf{x}_2$) if and only if all of the objectives of $\mathbf{x}_1$ are not inferior to that of $\mathbf{x}_2$ (i.e., $f_j(\mathbf{x}_1) \leq f_j(\mathbf{x}_2), \forall j \in \{1, 2, \cdots, M\}$) and $\mathbf{x}_1$ is better than $\mathbf{x}_2$ on at least one objective (i.e., $f_j(\mathbf{x}_1) < f_j(\mathbf{x}_2), \exists j \in \{1, 2, \cdots, M\}$). The symbol M denotes the number of objectives.

Pareto-optimal Solution: A feasible solution is commonly called Pareto-optimal if it cannot be dominated by any other feasible solution.

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

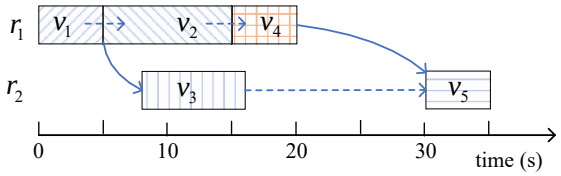
4. Algorithm Design

In a cloud platform, the scheduler is a middleware of the management system directly bridging tenants and cloud infrastructures. After a new workflow arrives, the scheduler analyzes the workflow's service requirements and acquires the status of cloud resources.

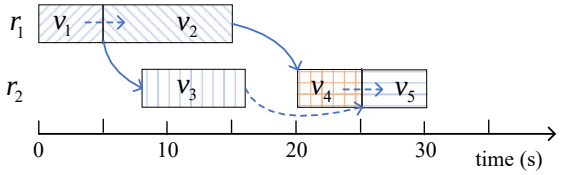
Then, the workflow scheduling algorithm will be triggered to decide the mappings from workflow tasks to resources, the execution order of tasks, and the start/finish time of cloud resources. The workflow scheduling for cloud platforms encounters various challenging factors such as scalable heterogeneous resources, various workflow structures, and multiple conflicting objectives. Searching for a set of compromise solutions is greatly significant. Based on the mainstream framework of multi-objective evolutionary algorithms, we design two problem-specific strategies: a critical task adjustment-based local search strategy and an idle gap reuse strategy.

4.1. Motivation Examples

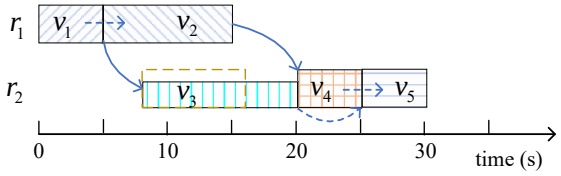
To visually illustrate the advantages of these two strategies, Figure 2 gives two examples. Suppose that the workflow in Figure 1 is to be scheduled and there are two available resources with the same configuration; the minimum execution times of the five tasks are $\{5, 10, 8, 5, 5\}$, and the data transfer time between tasks is summarized in Table 2. A feasible schedule based on the above assumptions is shown in Figure 2a. The solid directed edges indicate the data transfer constraints, and the dotted directed edges indicate that the data transfer time is negligible.



(**a**) A schedule



(**b**) A schedule by adjusting a critical task



(**c**) A schedule by adjusting CPU frequency for a non-critical task

Figure 2. Examples of motivation cases.

Table 2. Data transmission time (in seconds) among workflow tasks.

	t_1	t_2	t_3	t_4	t_5
t_1	–	5.0	3.0	–	–
t_2	–	–	5.0	–	–
t_3	–	–	–	–	5.0
t_4	–	–	–	–	10.0
t_5	–	–	–	–	–

Since the start time of task v_5 is determined by the output result of task v_4, task v_4 is a critical task. After adjusting the critical task v_4 to resource r_2, as illustrated in Figure 2b, the data transfer time from v_4 to v_5 is negligible. Then, the start/finish time of task v_5

is advanced, meaning that the makespan of the workflow is shortened. Furthermore, the working time of the two resources is shortened to reduce energy consumption. This example shows that the critical task adjustment-based local search strategy has the advantage of optimizing both the makespan and energy consumption of workflow execution. Furthermore, since task v_4 is constrained by the output result of task v_2, there is an idle time gap between v_3 and v_4. Given this fact, we adopt the DVFS technology to reduce the frequency/voltage of CPU processing task v_3 to reduce dynamic energy consumption, as shown in Figure 2c.

4.2. Main Framework

The proposed EMWSA follows the framework of classical multi-objective optimization algorithms and mainly includes three modules: initialization, reproduction of offspring population, and environmental selection. The main process of the proposed EMWSA is summarized in Algorithm 1.

Algorithm 1: Main Process of EMWSA

Data: Workflow G; Resource pool R; Population size N; Maximum number of function evaluations (MNE);
Result: A solution set;
1 $P \leftarrow$ Randomly initialize a population;
2 $FEs \leftarrow N$;
3 **while** $FEs < MNE$ **do**
4 $Q \leftarrow \emptyset$;
5 $P' \leftarrow$ Disturb the orders of solutions in P;
6 **for** $i = 1 \rightarrow N$ **do**
7 $\mathbf{x} \leftarrow$ Variation(P_i, P'_i);
8 $\mathbf{x} \leftarrow$ **Function** AdjustCriticalTask($\mathbf{x}$, G);
9 $\mathbf{x} \leftarrow$ **Function** AdjustCPUFrequency($\mathbf{x}$, G, R);
10 $q \leftarrow$ Calculate the objective vector using the formulas in (10) and (11) and obtain a new solution;
11 $Q \leftarrow Q \bigcup \{q\}$;
12 $FEs \leftarrow FEs + 1$;
13 **end**
14 $P \leftarrow$ EnvironmentalSelection($P \bigcup Q$, N);
15 **end**

As shown in Algorithm 1, the main inputs of the EMWSA are a workflow to be scheduled, resource pool, population size, and stop condition. When the EMWSA reaches the stop condition, it will output a set of non-dominated solutions.

In the initialization phase, a population is randomly generated (Line 1) and the number of used function evaluations (FEs) is recorded as N (Line 2). Before the FEs reach the maximum number of function evaluations (MNE), the EMWSA continuously iterates the remaining two phases: offspring population reproduction (Lines 4–13) and environment selection (Line 14). During offspring population reproduction, a set Q is initialized to store the offspring solutions (Line 4). Then, each solution is randomly combined with another one to generate an offspring solution (Line 7). Note that P_i denotes the i-th solution in population P. Next, Function $AdjustCriticalTask()$ is called to adjust some critical tasks to simultaneously optimize workflow's energy consumption and makespan, as described in Algorithm 2. After that, Function $AdjustCPUFrequency()$ is called to adjust the voltage/frequency of non-critical tasks to decrease energy consumption, as described in Algorithm 3. Since environmental selection is not the focus of this paper, we directly use the environmental selection operator in classical NSGA-II [40]. This approach first sorts the combined population into multiple non-domination levels and then conducts a crowding comparison procedure on the solutions in the last accepted levels.

4.3. Problem-Specific Optimization Strategies

For task v_i, we define its critical predecessor as the task whose data arrives at task v_i at the latest time among all of the predecessor tasks. Since the data transmission time between tasks executed on the same cloud resource can be negligible, adjusting the critical predecessor of a task to the same resource is promising to simultaneously reduce the energy consumption and makespan. This drives us to design a critical task adjustment-based local search strategy, as described in Algorithm 2.

Algorithm 2: Function AdjustCriticalTask($\vec{v}$, G, R)

Data: A decision vector **x**; the workflow G;
Result: A new decision vector **x**;
1 $C \leftarrow 0_{1\times n}$;
2 **for** $i = 1 \rightarrow n$ **do**
3 $\quad rt \leftarrow \max\{0, ft(v_b)\}$;
4 $\quad v_c \leftarrow \varnothing$;
5 $\quad$ **for** $v_p \in P(v_i)$ **do**
6 $\quad\quad ft \leftarrow 0$;
7 $\quad\quad dt \leftarrow ft(v_p) + \frac{I(x_p \neq x_i)\cdot w(e_{p,i})}{bw}$;
8 $\quad\quad$ **if** $dt > rt$ & $dt > ft$ **then**
9 $\quad\quad\quad ft \leftarrow dt$;
10 $\quad\quad\quad v_c \leftarrow v_p$;
11 $\quad\quad$ **end**
12 $\quad$ **end**
13 $\quad$ **if** $v_c \neq \varnothing$ **then**
14 $\quad\quad C(i) \leftarrow c$;
15 $\quad$ **end**
16 **end**
17 $U \leftarrow 1_{1\times|R|}$;
18 **for** $r_k \in R$ **do**
19 $\quad U(k) \leftarrow \frac{\sum_{v_i \in V_k} et(v_i)}{ft(r_k) - st(r_k)}$;
20 **end**
21 $U' \leftarrow 1_{1\times n}$;
22 **for** $t_i \in T$ **do**
23 $\quad U'(i) \leftarrow U(x_i)$;
24 **end**
25 $i^* \leftarrow$ Select a task using roulette according to $\frac{1/U'_i}{\sum_{i=1}^{n} 1/U'_i}$;
26 $j \leftarrow C(i^*)$;
27 $x_j \leftarrow x_{i^*}$;

From Algorithm 2, we can see that the main inputs of Function *AdjustCriticalTask*() are a decision vector and the workflow to be scheduled. This function consists of two stages: critical task identification (Lines 1–16) and adjustment (Lines 17–26). An array C is initialized to record the index of each task's critical precursor (Line 1). If the value of $C(i)$ is 0, it means that the task v_i has no critical precursor. For task v_i, its critical precursor v_c is defined as the one whose data arrives at task v_i at the latest time, and the arrival time is larger than the resource's ready time rt (Line 8). The symbol v_b denotes the task before task v_i on the same resource, and $ft(v_b)$ is the finish time of v_b. After identifying the critical precursors of each task, this function calculates the utilization rate of each resource (Lines 17–20). The symbol $U(k)$ is used to record the utilization rate of resource r_k; V_k denotes the set of tasks mapped to resource r_k; $ft(r_k)$ and $st(r_k)$ denote the finish time and start time of resource r_k, respectively. Next, the utilization rate of a resource is assigned to all of the tasks mapped to this resource (Lines 21–24). After that, according to the utilization rate

of the resources where the tasks are located, this function uses the roulette rule to select a task (Line 25), and adjusts its critical precursor to its mapped resource (Lines 26–27). In this way, this function is more likely to adjust critical tasks to resources with lower resource utilization.

Based on the fact that non-critical workflow tasks have certain slack times, we design an idle gap reuse strategy to lower the voltage/frequency of resources for energy conservation, as summarized in Algorithm 3.

Algorithm 3: Function AdjustCPUFrequency(**x**, G, R)

Data: A decision vector **x**;
Result: Decision of CPU frequency for each task;
1 **for** $v_i \in V$ **do**
2 $lft \leftarrow +\infty$;
3 **for** $v_s \in S(v_i)$ **do**
4 $ft \leftarrow st(v_s) - \frac{I(x_i \neq x_s) \cdot w(e_{i,s})}{bw}$;
5 **if** $ft < lft$ **then**
6 $lft \leftarrow ft$;
7 **end**
8 **end**
9 $lft \leftarrow \min\{lft, st(v_f)\}$;
10 **if** $lft > ft(v_i)$ **then**
11 $mec \leftarrow +\infty$;
12 $f^* \leftarrow \varnothing$;
13 $ft^* \leftarrow \varnothing$;
14 **for** $f_l \in F$ **do**
15 $ft'(v_i) \leftarrow$ Finish time of v_i under frequency f_l;
16 $ec \leftarrow$ Dynamic energy consumption of v_i under frequency f_l;
17 **if** $ft'(t_i) <= lft$ & $ec < mec$ **then**
18 $mec \leftarrow ec$;
19 $f^* \leftarrow f_l$;
20 $ft^* \leftarrow ft'(v_i)$;
21 **end**
22 **end**
23 **if** $f^* \neq \varnothing$ **then**
24 Assign the CPU frequency of resource r_{x_i} as f^* from $st(v_i)$ to ft^*;
25 **end**
26 **end**
27 **end**

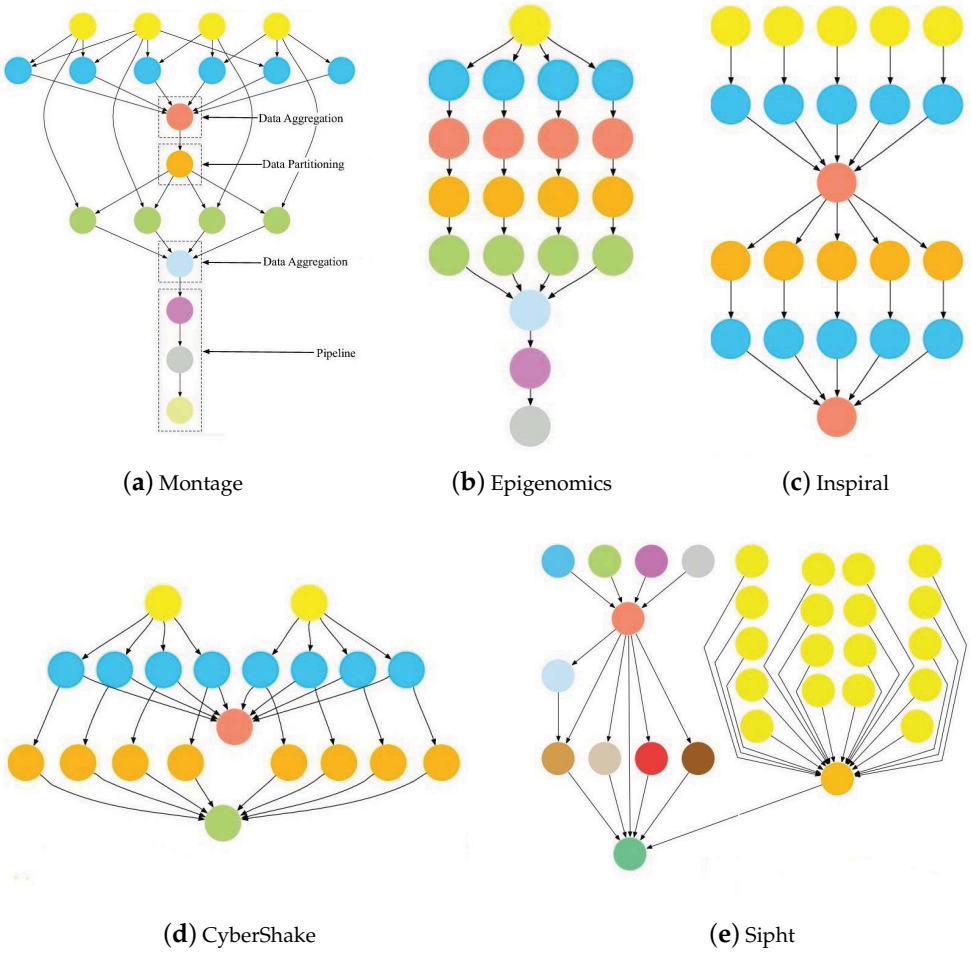
As illustrated in Algorithm 3, Function $AdjustCPUFrequency()$ lowers the execution frequency of non-critical tasks according to a decision vector about the mappings from tasks to resources. For a task v_i, its latest finish time lft is defined as the time point that does not affect the start of all of its successor tasks (Lines 3–8) and the task being executed after it (Line 9). The symbol v_f denotes the task being executed after task v_i. A task v_i is termed as non-critical if its latest finish time is greater than its finish time, that is $lft > ft(v_i)$. For non-critical tasks, this function finds the execution frequency f^* which minimizes the dynamic energy consumption mec and meets the constraint of the latest finish time (Line 17). The parameter ft^* records the corresponding finish time of the tasks. Next, the execution frequency and finish time of non-critical tasks are adjusted (Lines 24).

5. Performance Evaluation

In the context of real-world workflow traces and cloud platforms, this section evaluates the performance of the proposed EMWSA by comparing it with four representative baselines: EMS-C [35], SGA [41], GALCS [34], and MOELS [42].

5.1. Experimental Setups

Five different kinds of real-world workflows released by the Pegasus library have various model structures and have been widely employed to investigate the performance of workflow scheduling algorithms. We also employ these workflows to thoroughly test the proposal and the four existing algorithms, including Montage with 25, 50, 100, and 1000 tasks; Epigenomics with 24, 46, 100, and 997 tasks; Inspiral with 30, 50, 100, and 1000 tasks; Cybershake with 30, 50, 100, and 1000 tasks; and Sipht with 30, 60, 100, and 1000 tasks. Figure 3 gives the DAG examples for these five kinds of workflows with small-scale tasks. We can observe that these workflows pose complicated structures, including in-tree, out-tree, fork-join, pipeline, and mixture. For more details on these workflows, please refer to the Pegasus library repository at https://confluence.pegasus.isi.edu/display/pegasus (accessed on 1 October 2022).



(**a**) Montage (**b**) Epigenomics (**c**) Inspiral

(**d**) CyberShake (**e**) Sipht

Figure 3. DAG diagrams of workflows with about 30 tasks.

In the experiment, the power consumption and performance configurations of three different types of cloud resources are employed. Table 3 summarizes their relevant parameters.

Hypervolume [43] is a popular metric to measure the performance of multi-objective optimization approaches in terms of both convergence and diversity. The calculation of this metric does not require knowledge about problems' Pareto-optimal fronts, and is only a reference point. Since the Pareto-optimal front of the multi-objective workflow scheduling problem is unavailable and the reference point can be set according to the initialization population, this metric is suitable for testing multi-objective workflow scheduling algorithms. Assume $\mathbf{r} = \{r_1, r_2, \ldots, r_m\}$ is a reference point. For a population P, its hypervolume value

represents the hypervolume formed by reference point **r** and the P in the objective space and is computed as follows:

$$HV(P) = L(\bigcup_{p \in P}[f_1(p), r_1] \times [f_2(p), r_2] \cdots \times [f_m(p), r_m]), \quad (13)$$

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

For fairness, the population size of all 5 algorithms is set to 120; the maximum number of function evaluations is set according to the number of decision variables n, that is $n \times 3 \times 10^3$.

All five multi-objective workflow scheduling algorithms are written in MATLAB, and each experiment is repeated 30 times. The experimental environment mainly includes CPU (Intel(R) Xeon(R) Gold 6226R, 2.90 GHz), Memory (256.0 GB), Hard Disk (4.0 TB), Windows 10 operating system, and MATLAB 2020b.

Table 3. CPU configurations with respect to voltage/frequency pairs.

Levels	ADM Turion MT-34		AMD Opteron 2218		Intel Xeon E5450	
	Vol. (V)	Fre. (GHz)	Vol. (V)	Fre. (GHz)	Vol. (V)	Fre. (GHz)
0	1.20	1.80	1.30	2.60	1.35	3.00
1	1.15	1.60	1.25	2.40	1.17	2.67
2	1.40	1.40	1.20	2.20	1.00	2.33
3	1.20	1.20	1.15	2.00	0.85	2.00
4	1.00	1.00	1.10	1.80	–	–
5	0.80	0.80	1.05	1.00	–	–

Vol.: supply voltage to the CPU; Fre.: CPU frequency.

5.2. Comparison Results

Table 4 summarizes the comparison results of the 5 algorithms on 20 different workflows. These results include the mean and variance (in brackets) of the hypervolume values. Note that we use the Wilcoxon rank-sum test with a significance of 0.1 to identify the difference between the EMWSA and the baselines. The marks $-$ and $\approx$ represent the corresponding baseline performing significantly worse than and similar to the proposed EMWSA, respectively. The best results on each workflow are highlighted in bold.

As shown in Table 4, except for Sipht with 30 tasks, the proposed EMWSA generates higher hypervolume values than the four baselines. The three baselines (EMS-C, GALCS, and MOELS) only evolve the mappings from workflow tasks to resources, without considering the dynamic voltage frequency scaling technology. Different from these three baselines, the proposed EMWSA has two advantages. On the one hand, the EMWSA intelligently adjusts some critical tasks to the same resource as their successor tasks for simultaneously reducing workflows' energy consumption and makespan. On the other hand, the EMWSA searches for the optimal energy consumption of each non-critical task without affecting the operation of other tasks, so as to further reduce energy consumption. Although SGA employs dynamic voltage frequency scaling technology to save energy consumption, it does not have the ability to adjust critical tasks to reduce both the energy consumption and makespan. The comparison with SGA illustrates the effectiveness of the proposed critical task adjustment-based local search strategy.

On these workflows, the reference point for calculating the hypervolume is set based on the initial population and is far from the output populations; thus, the hypervolume values for each algorithm are high. On the Montage application with 25 tasks, the proposed EMWSA improves the hypervolume of algorithm EMS-C, SGA, GALCS, and MOELS by 8.33%, 7.63%, 27.68%, and 106.39%, respectively. Another interesting phenomenon is that, as the scale of workflows increases, the advantages of the proposed EMWSA become more apparent. Taking the Montage application as an example, the proposed EMWSA improves the hypervolume of SGA by 7.63% in a scenario of 25 tasks, while the EMWSA obtains 32.44% improvement in a scenario of 1000 tasks.

Table 4. Comparison results for the five algorithms on 20 workflows in terms of the hypervolume metric.

Workflows	n	EMS-C	SGA	GALCS	MOELS	EMWSA
Montage	25	1.068×10^6 (4.7×10^4) −	1.075×10^6 (3.6×10^4) −	9.062×10^5 (7.1×10^4) −	5.606×10^5 (2.5×10^5) −	**1.157×10^6 (3.7×10^4)**
	50	7.896×10^5 (8.8×10^4) −	7.766×10^5 (9.1×10^4) −	1.139×10^5 (1.1×10^5) −	2.669×10^4 (4.2×10^4) −	**7.951×10^5 (6.5×10^4)**
	100	2.011×10^6 (3.3×10^5) −	1.958×10^6 (3.5×10^5) −	0.000×10^0 (0.0×10^0) −	0.000×10^0 (0.0×10^0) −	**2.285×10^6 (3.1×10^5)**
	1000	3.534×10^7 (2.4×10^6) −	3.517×10^7 (2.6×10^6) −	0.000×10^0 (0.0×10^0) −	0.000×10^0 (0.0×10^0) −	**4.658×10^7 (5.8×10^6)**
Epigenomics	24	2.812×10^9 (2.0×10^7) −	2.796×10^9 (4.8×10^7) −	2.731×10^9 (3.9×10^7) −	2.352×10^9 (1.6×10^8) −	**2.825×10^9 (2.8×10^9)**
	46	4.378×10^9 (6.7×10^7) −	4.388×10^9 (4.9×10^7) −	4.205×10^9 (1.2×10^8) −	3.666×10^9 (3.2×10^8) −	**4.406×10^9 (5.5×10^7)**
	100	1.081×10^{11} (8.0×10^8) −	1.071×10^{11} (1.4×10^9) −	1.034×10^{11} (3.8×10^9) −	7.873×10^{10} (9.2×10^9) −	**1.088×10^{11} (7.2×10^7)**
	997	3.823×10^{12} (7×10^{10}) −	3.817×10^{12} (8×10^{10}) −	2.365×10^{12} (4×10^{11}) −	2.122×10^{12} (3×10^{11}) −	**4.033×10^{12} (7×10^{10})**
Inspiral	30	7.197×10^7 (2.7×10^6) −	7.346×10^7 (3.7×10^6) −	6.168×10^7 (4.9×10^6) −	3.900×10^7 (7.5×10^6) −	**8.664×10^7 (4.8×10^6)**
	50	5.515×10^7 (3.7×10^6) −	5.592×10^7 (3.4×10^6) −	4.092×10^7 (7.2×10^6) −	2.527×10^6 (4.5×10^6) −	**6.006×10^7 (2.8×10^6)**
	100	1.428×10^8 (1.8×10^6) −	1.416×10^8 (4.3×10^6) −	1.848×10^7 (6.4×10^6) −	8.216×10^6 (2.8×10^5) −	**1.482×10^8 (1.5×10^6)**
	1000	1.797×10^9 (2.4×10^7) −	1.932×10^9 (9.0×10^7) −	0.000×10^0 (0.0×10^0) −	0.000×10^0 (0.0×10^0) −	**2.360×10^9 (1.7×10^7)**
CyberShake	30	3.641×10^7 (3.2×10^6) −	1.978×10^7 (4.2×10^6) −	1.139×10^7 (4.1×10^6) −	7.417×10^5 (1.2×10^6) −	**6.146×10^7 (4.3×10^6)**
	50	7.614×10^7 (3.9×10^6) −	8.165×10^7 (6.9×10^6) −	6.560×10^7 (1.2×10^6) −	2.549×10^7 (9.2×10^6) −	**8.356×10^7 (3.8×10^6)**
	100	2.689×10^8 (9.3×10^6) −	2.726×10^8 (1.3×10^7) −	2.221×10^8 (2.5×10^7) −	9.230×10^7 (5.4×10^6) −	**2.922×10^8 (1.1×10^7)**
	1000	8.170×10^8 (4.0×10^7) −	7.876×10^8 (3.1×10^7) −	0.000×10^0 (0.0×10^0) −	3.579×10^7 (4.5×10^7) −	**8.263×10^8 (7.6×10^7)**
Sipht	30	**1.825×10^8 (1.2×10^6)** ≈	1.809×10^8 (6.7×10^4) −	1.784×10^8 (7.1×10^5) −	1.343×10^8 (8.4×10^5) −	1.824×10^8 (8.3×10^5)
	60	3.415×10^8 (2.0×10^6) −	3.413×10^8 (2.1×10^6) −	3.338×10^8 (1.9×10^6) −	2.859×10^8 (1.7×10^6) −	**3.444×10^8 (9.4×10^5)**
	100	3.435×10^8 (2.1×10^6) −	3.413×10^8 (1.5×10^6) −	3.328×10^8 (1.7×10^6) −	2.068×10^8 (3.1×10^7) −	**3.462×10^8 (1.3×10^6)**
	1000	6.137×10^9 (1.4×10^8) −	6.252×10^9 (8.8×10^7) −	2.941×10^9 (5.3×10^8) −	1.170×10^9 (6.9×10^8) −	**6.869×10^9 (9.5×10^7)**

To visually compare the convergence and diversity of the 5 algorithms (i.e., EMS-C, SGA, GALCS, MOELS, and EMWSA), we select their populations with the highest hypervolume values among 30 repeats on Montage, Epigenomics, Inspiral, CyberShake, and Sipht. Figure 4 draws these populations in the objective space.

As illustrated in Figure 4, the output populations of the proposed EMWSA have a better convergence and diversity when solving the problems derived from the five workflows. The better convergence of the EMWSA means that its solutions protrude to the origin in the objective space. The better diversity refers to its solutions covering a wider range. On Montage with 100 tasks, after the first objective value greater than 1.55×10^5, the proposed EMWSA and two baselines (EMS-C and SGA) have their own merits. When the first objective value is less than 1.42×10^5, the proposed EMWSA is obviously superior to the two baselines. Regarding the solutions obtained by GALCS, they are far dominated by that of the EMWSA. Although the MOELS achieves good results on the first objective, its results on the second objective are far worse than the EMWSA. These visual results are consistent with the higher hypervolume values of the EMWSA in Table 4. On the other four workflows, the five scheduling algorithms maintain good diversity. However, the proposed EMWSA shows a better convergence. This feature is particularly evident on Cybershake with 100 tasks. The results in Figure 4 visually reveal the superiority of the EMWSA in simultaneously optimizing energy consumption and makespan.

5.3. Trends of Hypervolume Values

To investigate the search characteristics of the five multi-objective workflow scheduling algorithms, Figure 5 shows the growth trends of their hypervolume values with the evolution process.

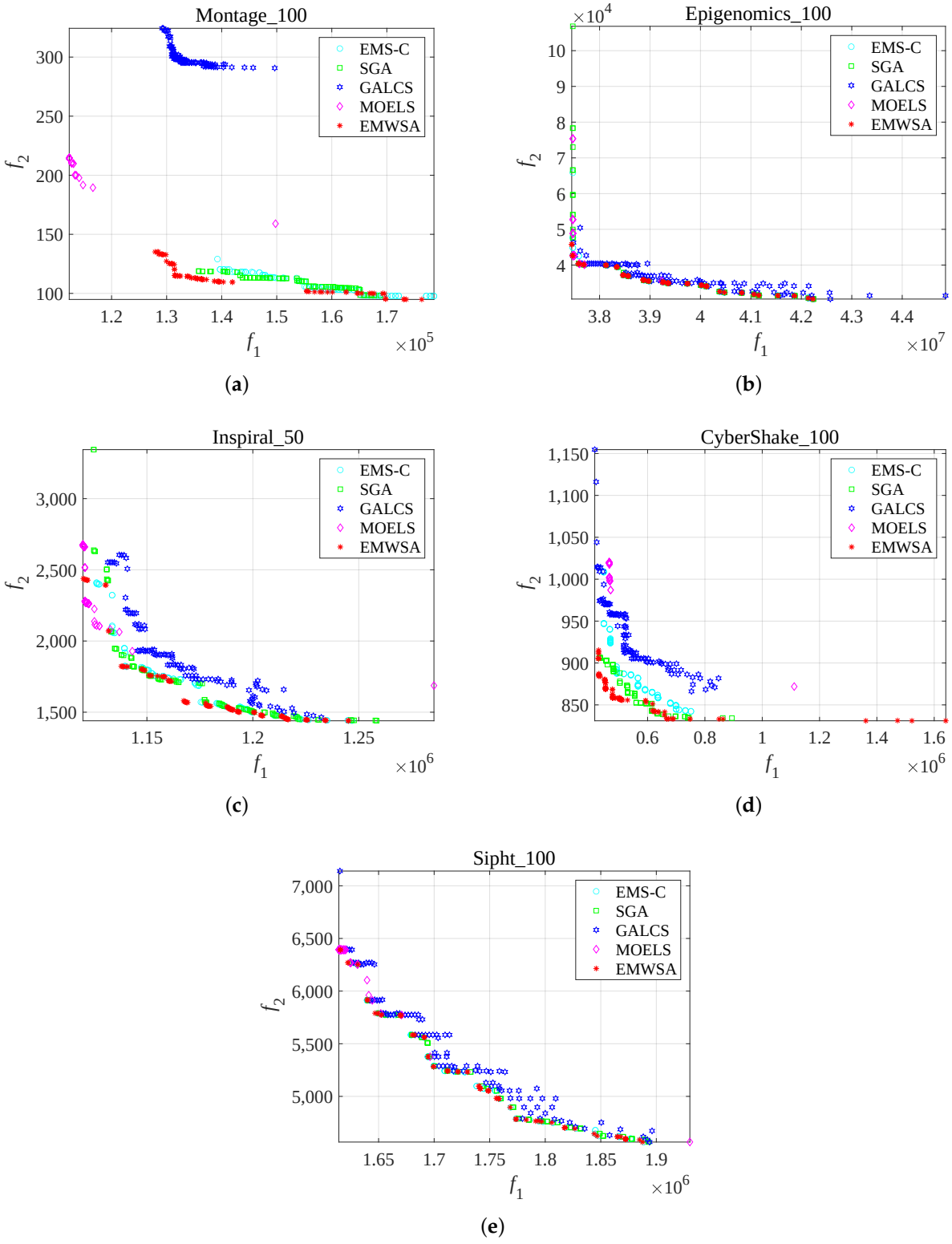


Figure 4. Output populations of the 5 algorithms on Montage, Epigenomics, Inspiral, CyberShake, and Sipht. (**a**) on Montage with 100 tasks; (**b**) on Epigenomics with 100 tasks; (**c**) on Inspiral with 50 tasks; (**d**) on CyberShake with 100 tasks; (**e**) on Sipht with 100 tasks.

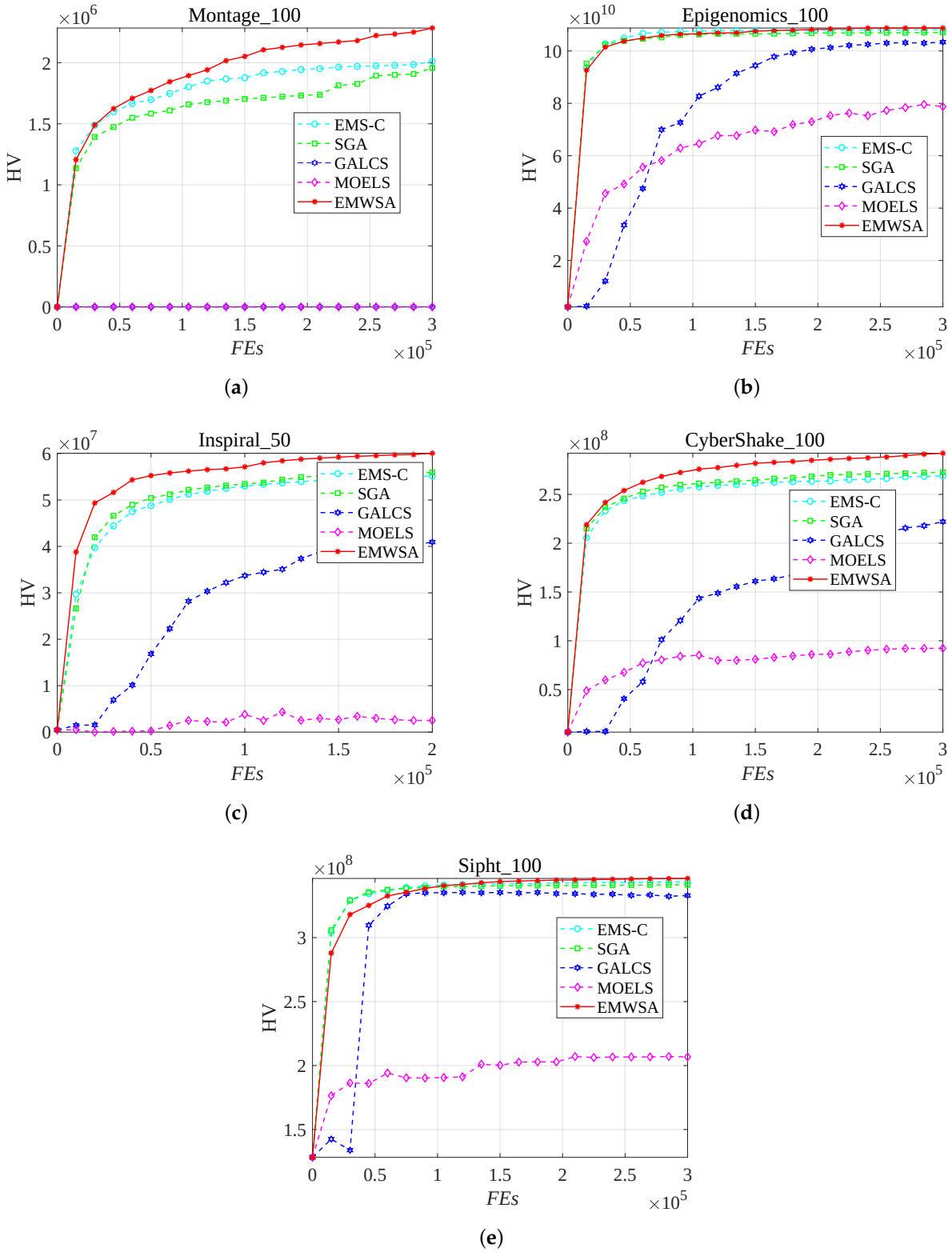


Figure 5. Change of hypervolume values with the advance of evolution. (**a**) on Montage with 100 tasks; (**b**) on Epigenomics with 100 tasks; (**c**) on Inspiral with 50 tasks; (**d**) on CyberShake with 100 tasks; (**e**) on Sipht with 100 tasks.

It is eye-catching in Figure 5 that the hypervolume values of the five scheduling algorithms grow quickly at the early stage. This can be summed up with the following facts. The scale of available resources in cloud platforms is very huge, and it is difficult to use these

resources properly by randomly initializing solutions. That is, workflow tasks are sparsely distributed on different resources, and the data transfer overheads among tasks causes considerable resource waste. In such a scenario, evolutionary algorithms can aggregate workflow tasks onto limited resources to simultaneously reduce energy consumption and makespan. This demonstrates that evolutionary algorithms have considerable potential in solving cloud workflow scheduling problem.

Figure 5 also illustrates that, with the deepening of the evolutionary search, the hypervolume values of the EMWSA still maintain a certain growth rate, while that of the other four baselines basically stops growing in the second half. Compared with the four baselines, the biggest feature of the EMWSA is that it embraces a critical task adjustment-based local search strategy. These comparison results demonstrate that the proposed local search strategy is conducive to jumping out of the local optimum, thus maintaining a strong search ability.

6. Conclusions and Future Work

This paper focuses on balancing the energy consumption and makespan of workflow executions running on cloud platforms, and mathematically formulates it as a multi-objective optimization problem. To resolve this challenging problem, this paper explores the problem's characteristics to tailor energy- and makespan-aware workflow scheduling algorithms with two new aspects. A critical task adjustment strategy is proposed to mitigate the negative impact of data transfer overheads among workflow tasks, simultaneously optimizing energy consumption and makespan. Furthermore, an idle gap reuse strategy is proposed to lower the execution CPU frequency of each non-critical task, so as to further reduce energy consumption. The performance of the proposal is evaluated by comparing it with four representative baselines in the context of twenty real-world workflows. Numerical results reveal that the proposal outperforms the four existing baselines in balancing energy consumption and makespan.

In addition to CPUs' energy consumption, other facilities in cloud platforms, such as storage and network equipment, are still significant energy consumption sources. In future work, we will comprehensively consider various sources of energy consumption, analyze their relationships, and develop systematic energy-saving technologies.

Author Contributions: Conceptualization and investigation, L.X. and F.H.; methodology, J.L. and Z.C.; validation, L.X. and F.H.; writing—original draft preparation, L.X.; writing—review and editing, J.L., Z.C. and F.H.; supervision, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Science and Technology Innovation Team of Shaanxi Province (2023-CX-TD-07), 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).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Armbrust, M.; Fox, A.; Griffith, R.; Joseph, A.D.; Katz, R.; Konwinski, A.; Lee, G.; Patterson, D.; Rabkin, A.; Stoica, I. A view of cloud computing. *Commun. ACM* **2010**, *53*, 50–58. [CrossRef]
2. Gill, S.S.; Buyya, R. A taxonomy and future directions for sustainable cloud computing: 360 degree view. *ACM Comput. Surv.* **2018**, *51*, 1–33. [CrossRef]

3. Dai, X.; Xiao, Z.; Jiang, H.; Alazab, M.; Lui, J.C.; Min, G.; Dustdar, S.; Liu, J. Task offloading for cloud-assisted fog computing with dynamic service caching in enterprise management systems. *IEEE Trans. Ind. Inform.* **2023**, *19*, 662–672. [CrossRef]
4. Zhang, J.; Liu, Y.; Li, Z.; Lu, Y. Forecast-Assisted Service Function Chain Dynamic Deployment for SDN/NFV-Enabled Cloud Management Systems. *IEEE Syst. J.* **2023**, 1–12. [CrossRef]
5. Lv, Z.; Chen, D.; Lv, H. Smart city construction and management by digital twins and BIM big data in COVID-19 scenario. *ACM Trans. Multimed. Comput. Commun. Appl.* **2022**, *18*, 1–21. [CrossRef]
6. Ye, R.; Liu, P.; Shi, K.; Yan, B. State damping control: A novel simple method of rotor UAV with high performance. *IEEE Access* **2020**, *8*, 214346–214357. [CrossRef]
7. Lv, Z.; Qiao, L.; Hossain, M.S.; Choi, B.J. Analysis of using blockchain to protect the privacy of drone big data. *IEEE Netw.* **2021**, *35*, 44–49. [CrossRef]
8. Yuan, H.; Bi, J.; Zhou, M.; Liu, Q.; Ammari, A.C. Biobjective task scheduling for distributed green data centers. *IEEE Trans. Autom. Sci. Eng.* **2020**, *18*, 731–742. [CrossRef]
9. Jones, N. How to stop data centres from gobbling up the world's electricity. *Nature* **2018**, *561*, 163–166. [CrossRef]
10. Li, M.; Tian, Z.; Du, X.; Yuan, X.; Shan, C.; Guizani, M. Power normalized cepstral robust features of deep neural networks in a cloud computing data privacy protection scheme. *Neurocomputing* **2023**, *518*, 165–173. [CrossRef]
11. Min, C.; Pan, Y.; Dai, W.; Kawsar, I.; Li, Z.; Wang, G. Trajectory optimization of an electric vehicle with minimum energy consumption using inverse dynamics model and servo constraints. *Mech. Mach. Theory* **2023**, *181*, 105185. [CrossRef]
12. Duan, Y.; Zhao, Y.; Hu, J. An initialization-free distributed algorithm for dynamic economic dispatch problems in microgrid: Modeling, optimization and analysis. *Sustain. Energy Grids Netw.* **2023**, *34*, 101004. [CrossRef]
13. Lee, Y.C.; Zomaya, A.Y. Energy conscious scheduling for distributed computing systems under different operating conditions. *IEEE Trans. Parallel Distrib. Syst.* **2011**, *22*, 1374–1381. [CrossRef]
14. Masdari, M.; Zangakani, M. Green cloud computing using proactive virtual machine placement: challenges and issues. *J. Grid Comput.* **2020**, *18*, 727–759. [CrossRef]
15. Li, Z.; Ge, J.; Hu, H.; Song, W.; Hu, H.; Luo, B. Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds. *IEEE Trans. Serv. Comput.* **2018**, *11*, 713–726. [CrossRef]
16. 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]
17. 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]
18. 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]
19. Hafsi, H.; Gharsellaoui, H.; Bouamama, S. Genetically-modified Multi-objective Particle Swarm Optimization approach for high-performance computing workflow scheduling. *Appl. Soft Comput.* **2022**, *122*, 108791. [CrossRef]
20. Tian, J.; Hou, M.; Bian, H.; Li, J. Variable surrogate model-based particle swarm optimization for high-dimensional expensive problems. *Complex Intell. Syst.* **2022**, 1–49. [CrossRef]
21. 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]
22. Zhang, K.; Wang, Z.; Chen, G.; Zhang, L.; Yang, Y.; Yao, C.; Wang, J.; Yao, J. Training effective deep reinforcement learning agents for real-time life-cycle production optimization. *J. Pet. Sci. Eng.* **2022**, *208*, 109766. [CrossRef]
23. Feng, Q.; Feng, Z.; Su, X. Design and simulation of human resource allocation model based on double-cycle neural network. *Comput. Intell. Neurosci.* **2021**, *2021*, 7149631. [CrossRef] [PubMed]
24. Hussain, M.; Wei, L.F.; Rehman, A.; Abbas, F.; Hussain, A.; Ali, M. Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers. *Future Gener. Comput. Syst.* **2022**, *132*, 211–222. [CrossRef]
25. Bharany, S.; Badotra, S.; Sharma, S.; Rani, S.; Alazab, M.; Jhaveri, R.H.; Gadekallu, T.R. Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy. *Sustain. Energy Technol. Assess.* **2022**, *53*, 102613. [CrossRef]
26. Medara, R.; Singh, R.S.; Sompalli, M. Energy and cost aware workflow scheduling in clouds with deadline constraint. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e6922. [CrossRef]
27. Pan, L.; Liu, X.; Jia, Z.; Xu, J.; Li, X. A Multi-objective Clustering Evolutionary Algorithm for Multi-workflow Computation Offloading in Mobile Edge Computing. *IEEE Trans. Cloud Comput.* **2021**. [CrossRef]
28. 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]
29. Mohammadzadeh, A.; Masdari, M.; Gharehchopogh, F.S.; Jafarian, A. A hybrid multi-objective metaheuristic optimization algorithm for scientific workflow scheduling. *Clust. Comput.* **2021**, *24*, 1479–1503. [CrossRef]
30. 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]
31. Paknejad, P.; Khorsand, R.; Ramezanpour, M. Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment. *Future Gener. Comput. Syst.* **2021**, *117*, 12–28. [CrossRef]

32. Peng, K.; Zhu, M.; Zhang, Y.; Liu, L.; Zhang, J.; Leung, V.C.; Zheng, L. An energy-and cost-aware computation offloading method for workflow applications in mobile edge computing. *EURASIP J. Wirel. Commun. Netw.* **2019**, *2019*, 207. [CrossRef]
33. Tarafdar, A.; Debnath, M.; Khatua, S.; Das, R.K. Energy and makespan aware scheduling of deadline sensitive tasks in the cloud environment. *J. Grid Comput.* **2021**, *19*, 1–25. [CrossRef]
34. Xia, X.; Qiu, H.; Xu, X.; Zhang, Y. Multi-objective workflow scheduling based on genetic algorithm in cloud environment. *Inf. Sci.* **2022**, *606*, 38–59. [CrossRef]
35. 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]
36. 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]
37. Topcuoglu, H.; Hariri, S.; Wu, M.Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. Parallel Distrib. Syst.* **2002**, *13*, 260–274. [CrossRef]
38. 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]
39. 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]
40. 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]
41. Sathya Sofia, A.; GaneshKumar, P. Multi-objective task scheduling to minimize energy consumption and makespan of cloud computing using NSGA-II. *J. Netw. Syst. Manag.* **2018**, *26*, 463–485. [CrossRef]
42. 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]
43. 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]