

Article

HICA: A Hybrid Scientific Workflow Scheduling Algorithm for Symmetric Homogeneous Resource Cloud Environments

Liang Hu, Xianwei Wu and Xilong Che *

School of Computer Science and Technology, Jilin University, Changchun 130012, China

* Correspondence: chexilong@jlu.edu.cn

Abstract: With the increasing volume of scientific computation data and the advancement of computer performance, scientific computation is becoming more dependent on the powerful computing capabilities of cloud computing. On cloud platforms, tasks in scientific workflows are assigned to computational resources and executed according to specific strategies. Therefore, workflow scheduling has become a key factor affecting efficiency. This paper proposes a hybrid scientific workflow scheduling algorithm, HICA, to address the scheduling problem of scientific workflows in symmetric homogeneous cloud environments with optimization goals of makespan and cost. HICA combines the Imperialist Competitive Algorithm (ICA) with the HEFT algorithm, integrating HEFT into the initial population of the ICA to accelerate the convergence of the ICA. Experimental results show that the proposed hybrid approach outperforms other algorithms in real-world workflow applications. Specifically, when the workflow scale is 100, the average improvements in makespan and cost are 133.89 and 273.33, respectively; when the workflow scale is 1000, the improvements are 371.62 and 9178.98. The scheduling results for the Earth System Model parameter tuning workflow show that compared to the scenario without using a scheduling algorithm, the makespan and cost were improved by 13% and 21%, respectively.

Keywords: cloud computing; scientific workflow; workflow scheduling; Imperialist Competitive Algorithm; hybrid algorithm



Academic Editor: Hsien-Chung Wu

Received: 6 January 2025

Revised: 31 January 2025

Accepted: 8 February 2025

Published: 12 February 2025

Citation: Hu, L.; Wu, X.; Che, X. HICA: A Hybrid Scientific Workflow Scheduling Algorithm for Symmetric Homogeneous Resource Cloud Environments. *Symmetry* **2025**, *17*, 280. <https://doi.org/10.3390/sym17020280>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Scientific workflows are a process in complex scientific research and experiments where data collection, processing, qualitative analysis, and result visualization are combined on-demand, matching service selection and network resource mapping [1]. As scientific research continues to deepen, scientific workflows are becoming increasingly complex, requiring higher computational power to operate smoothly. The rise of cloud computing has just provided the computational capacity for the execution of scientific workflows. Therefore, more and more scientific computations are being executed in the form of workflows on cloud platforms. Symmetric homogeneous cloud environment due to their simple structure and strong scalability are widely used for executing data-intensive scientific workflows.

Compared to other HPC platforms, cloud computing offers more flexible services. According to the type of provided capability, the services of cloud computing are broadly divided into three categories: Infrastructure-as-a Service (IaaS), Platform-as-a-Service (PaaS), and Software as-a-Service (SaaS) [2]. Users can more freely configure resources by adjusting system parameters related to performance and storage, making it easy to consume all available resources [3]. In this situation, when users execute scientific workflows on cloud platforms, they must consider not only the completion of tasks but also the needs

related to time, resource usage, and cost. Therefore, through efficient scientific workflow scheduling algorithms, finding the mapping of workflow tasks to different resource nodes to satisfy different quality of service (QoS) requirements of users has become a critical issue in the execution of workflows on a cloud computing platform.

Some list-based scheduling methods analyze the relationships between nodes, set priorities for nodes based on weights [4], and then sort the workflow nodes according to these weights. Although this approach is easy to execute, as the number of workflow nodes and available resources increases, this type of algorithm can become time consuming because it needs to traverse the execution status of workflow nodes on each available resource. Moreover, workflow scheduling is generally considered an NP-complete problem, making it difficult to obtain the best solution within a limited timeframe. Considering the ever-increasing scale and complexity of modern workflows, there is a need for more efficient scheduling algorithms.

The essence of workflow scheduling is to construct the mapping between task nodes and resources [5] under which the workflow not only executes successfully but also meets the user's QoS parameters. When QoS requirements are clearly quantified, the problem can be transformed into a parameter optimization problem, which is aimed at optimizing scheduling decisions to improve system performance such as minimizing makespan and minimizing cost. To address the complexity and diverse optimization objectives in workflow scheduling, many parameter optimization algorithms have been proposed and applied in this field. These optimization algorithms build mathematical models and adjust scheduling parameters to seek optimal or near-optimal scheduling solutions. Common parameter optimization algorithms include Particle Swarm Optimization (PSO) [6] and Genetic Algorithms (GAs) [7]. By exploring the solution space, these algorithms can optimize scheduling decisions while satisfying multiple constraints, aiming to minimize workflow performance metrics and simultaneously improve resource utilization.

However, the optimization efficiency of these algorithms is usually closely related to the quality of the initial state. A better initial state, which can reduce the number of iterations, makes the algorithm converge more quickly. In this paper, a hybrid workflow scheduling algorithm HICA is proposed. We use the Imperialist Competitive Algorithm (ICA) to find a scheduling mapping that meets the requirements. HEFT is introduced, and the scheduling sequence it generates is used as one of the initial states for the ICA algorithm. Moreover, we design an encoding scheme for the HICA algorithm to solve the scheduling problem such that individuals and iterations in the ICA can represent scheduling maps and the quality of different scheduling maps. We build a symmetric homogeneous resource cloud environment using WorkflowSim and conduct experiments using real-world workflow applications as scheduling targets. The results indicate that HICA outperforms other methods in workflow scheduling with significant advantages in both makespan and cost.

The structure of this paper is as follows: Section 2 discusses related work on workflow scheduling algorithms. We introduce the workflow scheduling problem in Section 3. Section 4 describes the proposed hybrid workflow scheduling algorithm HICA. Section 5 presents the results of our experiments. Finally, Section 6 outlines the general conclusions and discusses future work.

2. Related Works

2.1. Workflow Management Tools

Since the introduction of scientific workflow technology into scientific computing, many research efforts have aimed to provide researchers with more convenient tools for managing scientific workflows. For example, Pegasus [8] allows users to execute tasks

across multiple computational resources and is capable of handling large-scale workflows, and it has been applied across a broad array of scientific fields, such as seismology, bioinformatics, astronomy, etc. Kepler [9] offers a powerful graphical workbench, allowing them to freely select components based on functional requirements. It has been widely applied in projects in different fields, including oceanography, data management, and biology. Taverna [10] supports workflow-based biological experiments, allowing users to design and monitor workflow information on both web platforms and Android mobile devices. Askalon [11] provides fault-tolerance handling and other functionalities for workflows in a grid environment. However, most of these studies focus on enhancing user experience and ease of operation; they have largely overlooked the issue of workflow scheduling.

2.2. Workflow Scheduling Algorithm

Many parameter optimization algorithms are inspired by natural phenomena and designed to address complex optimization challenges [12]. Recently, these algorithms have gained significant attention in the context of workflow scheduling applications. Yu and Buyya [13] introduced the application of a Genetic Algorithm (GA) to address the workflow scheduling problem, taking into account both deadline and budget constraints. Chen et al. [14] proposed a grid workflow scheduling method based on Ant Colony Optimization (ACO). Rodriguez and Buyya [15] employ a Particle Swarm Optimization (PSO)-based workflow scheduling approach to minimize costs while ensuring deadline constraints are met. Bilgaiyan et al. [16] demonstrated that CSO can be used to solve the problem of cost minimization in workflow scheduling and also guarantees an equitable distribution of load across the available resources. Sagnika et al. [17] applied the Bat Algorithm (BA) to schedule workflows in a cloud environment. Rehani and Garg [18] proposed the multi-objective NSGA-II based scheduling algorithm for workflow applications with the aim to optimize three conflicting criterion simultaneously. Rehani and Garg [18] implemented workflow scheduling using NSGA-II to address multiple conflicting criteria. These studies have demonstrated that for workflow scheduling, parameter optimization algorithms can achieve scheduling that meets different QoS requirements. Recently proposed algorithms such as red fox optimizer [19], fly fox optimizer [20], polar bear optimization [21] and the mayfly optimization algorithm [22] have demonstrated their effectiveness in many fields and show great potential in the workflow scheduling domain. Related research is underway to apply these methods to workflow scheduling problems.

Building on this, some studies have attempted to combine different optimization algorithms, taking advantage of their individual strengths to further enhance the efficiency and quality of scheduling. Verma and Kaushal [23] proposed a hybrid Particle Swarm Optimization (HPSO) scheduling method, which can provide Pareto-based solutions for multi-objective problems. Mohammadzadeh et al. [24] proposed an algorithm that hybridizes the ant lion optimizer (ALO) with Sine Cosine Algorithm (SCA) and applies it to solve the multi-objective workflow scheduling problem. Ramathilagam and Vijayalakshmi [25] proposed EAFSAIPR with the HECC method, which combines the Enhanced Artificial Fish Swarm Algorithm with IaaS Cloud Partial Critical Path and Hyper Elliptic Curve Cryptography. Aziza and Krichen [26] proposed a hybrid approach using genetic algorithms and HEFT to optimize cloud workflow scheduling. Mohammadzadeh and Masdari [27] presented a hybrid algorithm HGSOA-GOA to address the multi-objective scheduling problem. Kakkottakath Valappil Thekkepurayil et al. [28] proposed a hybrid scheduling algorithm that effectively combines the advantages of ant lion optimization (ALO) and popular Particle Swarm Optimization. Shirvani [29] presented a discrete Particle Swarm Optimization-based hybrid scheduling algorithm, which can avoid becoming trapped in local optima. Mikram et al. [30] used the Hybrid HEFT-PSO-GA algorithm

(HEPGA) to provide a task scheduling sequence for the data center that minimizes the makespan. Choudhary et al. [31] proposed a hybrid scheduling method combining GSA and HEFT to address the makespan and cost issues. Sharma et al. [32] proposed the Cuckoo Search Flower Pollination Algorithm (CSFPA); the Cuckoo Search and Flower Pollination Algorithm are used in different search phases to achieve scheduling optimization for both time and cost. Table A1 lists the scheduling objectives and research methods of these studies.

3. Problem Definition

This paper addresses workflow scheduling problem with the goal of minimizing both computational costs and makespan. We will introduce the workflow structure and definition of the QoS requirements in the remaining part of this section.

3.1. Workflow Structure

To more clearly represent the dependencies between tasks, workflows are typically depicted as a directed acyclic graph (DAG). A DAG is typically presented as $G(T, E)$, where T denotes the tasks with dependencies, and E represents the edge connecting them. Figure 1 shows a DAG workflow with eight tasks. In this figure, the task set T contains eight tasks, there are $T_1 - T_8$ and there are nine edges in the edge set E . The direction of the edges indicates the flow of data and dependency relationships between the nodes. For example, $E(2, 6)$ represents that the the direction of data transfer is from node 2 to node 6. This means that node 6 must wait until the task on node 2 is completed, and the required data are transferred to node 6, before it can start executing its own task to satisfy the dependency relationship. For a child node, it can only meet the execution condition when it has received input data from all its parent nodes. For example, task 6 must wait until tasks 2, 3 and 4 have all completed their execution and transferred their output data to task 6 before it satisfies the condition for execution.

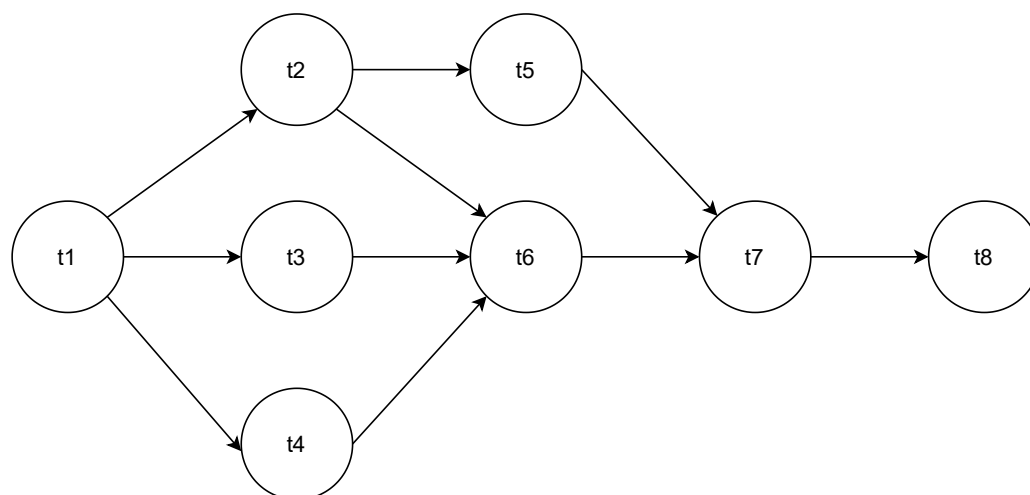


Figure 1. A workflow sample organized with DAG.

To more clearly represent the dependency relationships between nodes, we describe the workflow node dependencies in Figure 1 using a matrix format, as shown in Figure 2. $E(t_m, t_n) = 1$ means the presence of an edge from task m to task n . When considering the transmitted data, we use the edge weights to represent the amount of data output by the parent node and received by the child node. For example, if $E(t_3, t_6) = 13$, it means the data transformed from task 3 to task 6 is equal to 13.

	Child task							
	t1	t2	t3	t4	t5	t6	t7	t8
Parent task	t1	0	1	1	1	0	0	0
	t2	0	0	0	0	1	1	0
	t3	0	0	0	0	0	1	0
	t4	0	0	0	0	0	1	0
	t5	0	0	0	0	0	0	1
	t6	0	0	0	0	0	0	1
	t7	0	0	0	0	0	0	1
	t8	0	0	0	0	0	0	0

Figure 2. A matrix to describe the relation between the tasks in Figure 1.

3.2. Time and Cost Calculation

In this section, we will introduce the calculation methods for makespan and cost generated during the execution of a DAG-based workflow. Makespan refers to the total duration needed to complete a workflow, from the initiation of the first task to the completion of the final task, and it is a key performance metric in workflow scheduling. The expression is as shown in Equation (1):

$$Makespan = \text{Max}\{CT_m\}, 1 \leq m \leq n \quad (1)$$

where CT_m represents the completion time of task i and n represents the total number of workflow tasks.

The cost is described as follows:

$$Cost = \sum_{p=1}^n (CT(p, q) - BT(p, q)) * price_q \quad (2)$$

where n is the number of task nodes in the workflow. $CT(p, q)$ represents the completion time of task p executed over resource q and $BT(p, q)$ represents the beginning time of task p executed over resource q . $price_q$ denotes the cost of resource q per unit time.

The BT and CT are shown in Equations (3) and (4):

$$BT(p, q) = \max(T_q, \max_{k \in pred_p} \frac{data}{bd}) \quad (3)$$

$$CT(p, q) = BT(p, q) + runtime(p, q) \quad (4)$$

where T_q represents the time at which resource q finishes all previous tasks and is ready for the next task. $pred_p$ means the predecessor node of task i . $data$ and bd represent the amount of data transmitted and the bandwidth, respectively. Their ratio represents the transmission time. It is important to note that if tasks k and p are executed on the shared resource, there is no transmission time. $runtime$ refers to the execution time of task p on resource node q .

4. Proposed HICA Algorithm

In this paper, we propose HICA, which is a hybrid workflow scheduling algorithm. We combine the HEFT algorithm and use it as one of the initial empires. This section provides a detailed explanation of the implementation process of the HICA algorithm.

4.1. Encoding

Encoding modeling plays a key role in solving scheduling problems [15]. To define the scheduling problem and solve it using ICA, we propose the following encoding: each solution is represented by an array with its length corresponding to the number of task nodes in the workflow. The array index represents the task node number, while the array content indicates the resource allocated to each task node. This encoding guarantees that each task is allocated to a single virtual machine instance, ensuring the scheduling of all tasks in the task pool, while a virtual machine instance can be selected by multiple tasks. It is important to note that not all virtual machine instances must be used in every solution. An example of the encoded array based on the workflow in Figure 1 is shown in Figure 3.

Task	t1	t2	t3	t4	t5	t6	t7	t8
Resource	2	4	1	3	4	4	1	3

Figure 3. An example of the encoding (based on Figure 1).

4.2. Fitness Function

To address the workflow scheduling problem by ICA, it is essential not only to properly model and encode the problem but also to design a fitness function that meets the scheduling requirements. Our objective is to simultaneously minimize computation time and cost, requiring a fitness function that effectively combines these two goals. The function should intuitively reflect the quality of the current scheduling scheme, where better scheduling results are linked to higher performance scores, and poorer results are linked to lower performance scores. The specific definition is as follows:

$$F(i) = \alpha \frac{1}{makespan_i} + (1 - \alpha) \frac{1}{cost_i}, \alpha \in (0, 1) \quad (5)$$

where $makespan_i$ and $cost_i$ represent the time and cost required to complete the i th scheduling sequence, respectively. α is the weight parameter to balance the time fitness function and cost function.

4.3. Generate Initial Countries

Algorithm 1 describes the process that generate the initial countries. In line 2, the execution result of the HEFT algorithm serves as one of the initial countries. For line 4, the calculation method for empires and colonies is as follows:

$$cost_i = f(country_i) \quad (6)$$

$$C_n = cost_n - \max(cost) \quad (7)$$

$$p_n = \left| \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \right| \quad (8)$$

$$N.C_n = \text{round}(p_n * N_{col}) \quad (9)$$

where $cost_i$ represents the fitness function value of the i th country. $\maxcost$ is the optimal fitness function value among all countries. p_n represents the normalized “power” size of each empire. $N.C$ represents the quantity of colonies for every empire. round represents a rounding function that rounds to the nearest integer. N_{imp} and N_{col} represent the count of empires and colonies, respectively. A selection of the most optimal countries from a total of N countries is made to form empires, while the remaining countries become colonies.

The normalized ‘power value’ of each empire is then calculated based on the fitness function values of the empire itself and its initial colonies. Subsequently, the remaining colonies are allocated to each empire proportionally according to their normalized power values.

Algorithm 1 Generate Initial Empires

Input: The number of country n .

Output: n scheduling sequences as the initial countries.

- 1: Randomly generate $n - 1$ countries.
 - 2: Add the country obtained by HEFT.
 - 3: Calculate fitness function value for each country.
 - 4: Sort the country based on their function value and determine empires and colonies.
-

4.4. Colonies Assimilation

In reality, empires, in order to better control their colonies, promote their own ideas, customs, and other aspects to the colonies, which is a process known as assimilation. In ICA, colonies also have an operation that moves them closer to the empire, simulating this assimilation process. Traditional ICA controls the gradual movement of colonies toward the imperial solution using a distance-based approach. However, in this paper, considering the specific attributes of the workflow scheduling problem, we adopt a discrete colony assimilation method that makes the structure of each colony similar to the empire’s structure in certain aspects, and this process occurs with a certain probability.

The colony assimilation operation is shown in Figure 4 and is implemented as follows:

1. Some elements in the colony are selected with a certain probability (there are 1, 4, and 6 in this picture).
2. The selected elements are directly copied from the empire’s corresponding elements.
3. The other elements in colony remain unchanged (there are 2, 3, 5, 7 and 8 in this picture).

	t1	t2	t3	t4	t5	t6	t7	t8
empire	1	3	2	3	4	2	3	4
colony	2	4	1	3	4	4	1	3
new colony	1	3	1	3	4	2	1	3

Figure 4. An example of a colony assimilation operation (based on Figure 1). The yellow cells represent the elements that perform the operation in the colony assimilation process.

4.5. Colonies Revolution

A colony enhances its strength through internal revolution to break free from imperial rule. In ICA, a similar operation exists for colonies, where two elements are randomly selected and their corresponding values are swapped. Should the new colony demonstrate superior performance compared to the existing one, the latter is supplanted by the former. This process is illustrated in Figure 5.

	t1	t2	t3	t4	t5	t6	t7	t8
colony	2	4	1	3	4	4	1	3
new colony	2	1	1	3	4	4	4	3

Figure 5. An example of a colony revolution operation (based on Figure 1). The yellow cells represent the elements that perform the operation in the colony revolution process.

4.6. Empire Update

Within an empire, situations may arise where a colony's power closely rivals that of the imperialist. Following processes of assimilation and revolution, certain colonies might achieve a state superior to that of the imperialist, as indicated by a more favorable fitness function value. When this occurs, a role reversal takes place: the colony ascends to the position of imperialist, while the former imperialist is relegated to the status of a colony. Subsequently, the colonies persist in undergoing assimilation and revolution under the governance of the newly established imperialist.

4.7. Imperial Competition

In addition to the conflict between empires and colonies, similar to real-life competition and plunder conducted by imperial groups to expand their influence, in the ICA algorithm, competition also occurs between different imperial groups. In the competition mechanism, each imperial group attempts to control the weaker colonies of the weaker imperial groups, making the strong stronger and the weak weaker, until only one imperial group remains. The imperial competition mechanism simulates the process in which stronger empires occupy and control the colonies of weaker empires in real society. First, the total cost function value of the empire, representing its "power", needs to be calculated.

$$T.C._n = cost_n + \zeta * \frac{\sum_{i=1}^{N.C_n} f(col_i)}{N.C_n} \quad (10)$$

where $T.C._n$ denotes the power value of the n -th empire, $cost_n$ is the fitness function value of the empire itself, and ζ is a coefficient that limits the total cost value of the entire imperial group, ranging between 0 and 1. A higher value of ζ corresponds to a greater power of the imperial collective and an enhanced influence thereof. The numerator encapsulates the sum of the fitness function values across all colonies, and the denominator is the number of colonies. The entire fraction represents the average power value of the empire's colonies.

The power value of each imperial group is determined according to Equation (10). During imperial competition, the most vulnerable empire is redistributed among other, more dominant empires with each imperial group competing for the colony based on the probability of occupation. Each imperial group has the opportunity to capture the weakest colony, so the weakest colony will not necessarily belong to the strongest imperial group. Of course, the stronger the imperial group, the greater the probability of capturing the colony.

The process of empire competition for colonies is shown in Algorithm 2. This process is similar to generating initial countries (Section 4.3). First, the normalized power value of each empire is calculated:

$$N.T.C_i = \max\{T.C\} - T.C_i \quad (11)$$

where $\max T.C$ represents the highest power value among all empires. Then, the probability P_i is calculated based on the normalized power value:

$$P_i = \left| \frac{N.T.C_i}{\sum_{j=1}^{N_{imp}} N.T.C_j} \right| \quad (12)$$

We form a vector P_i with each P_i , and it is clear that the dimension of this vector corresponds to the number of empires N_{imp} :

$$P = [p_1, p_2, \dots, p_{N_{imp}}] \quad (13)$$

The vector U has the same dimension as the vector P , and its elements follow a uniform distribution between 0 and 1, as in Equation (14):

$$U = [u_1, u_2, \dots, u_{N_{imp}}], \text{ where } u_1, u_2, \dots, u_{N_{imp}} \sim U(0, 1) \quad (14)$$

Subsequently, a vector D is constructed as delineated in Equation (15):

$$D = P - U = [D_1, D_2, \dots, D_{N_{imp}}] = [p_1 - u_1, p_2 - u_2, \dots, p_{N_{imp}} - u_{N_{imp}}] \quad (15)$$

The empire corresponding to the largest element in the vector D will acquire the smallest colony of the weakest empire.

Algorithm 2 Empire competition for colonies.

- 1: **for** Each empire **do**
 - 2: Calculate normalized power value $N.T.C.$
 - 3: Calculate empire's probability P_i
 - 4: **end for**
 - 5: Generate vector P according to each P_i
 - 6: Generate vector R and calculate vector $D = P - R$.
 - 7: The empire with the greatest value in D acquires the colony.
-

4.8. Empire Perishes

The competition between empires causes the stronger empires to become increasingly powerful by acquiring the colonies of weaker empires, while the number of colonies of weaker empires continuously decreases. When an empire loses all its colonies, it is defeated. As empires fall, eventually only one dominant ruler remains, and the algorithm terminates.

5. Experiments and Results

This section evaluates the proposed algorithm by implementing the methodology outlined in Section 4 and comparing its performance against other state-of-the-art algorithms. The experimental setup, simulation environment, and detailed results are presented in the following sections.

5.1. Simulation Setup

The selected experimental tool is WorkflowSim [33], which is an extension of CloudSim [34]. It addresses the issue that neglecting system overheads and failures during the simulation of scientific workflows can result in considerable inaccuracies when estimating workflow runtime. The compute resource configurations are based on [26] and summarized in Table 1.

To validate the effectiveness of HICA, we selected several widely used algorithms for comparison. These algorithms are broadly applied in the field of workflow scheduling. They are PEFT-GA [35], GA-PSO [36], Greedy-Ant [37] and ICA [38]. All experiments were

conducted on a computer equipped with a 3.7 GHz Intel Core i3-10105 CPU and 16 GB of 2666 MHz RAM.

Table 1. Experimental configuration and cost parameters for virtual machines (VMs).

VM Parameters	Value	Cost Parameters	Value
Number of VMs	20	Processing usage cost	3.0
RAM (MB)	512	Memory usage cost	0.05
MIPS	1000	Storage usage cost	0.1
Bandwidth	1000	Bandwidth usage cost	0.1

5.2. Workflow Applications

Some studies evaluate the performance of scheduling algorithms by generating random workflows. To demonstrate the capability of HICA in addressing real-world engineering problems, we adopted real scientific workflow applications, whose structures are shown in Figure 6. Specifically, CyberShake is used for earthquake prediction, Montage is applied in astronomy, LIGO is used in physics, while SIPHT and Epigenomics are applied in bioinformatics.

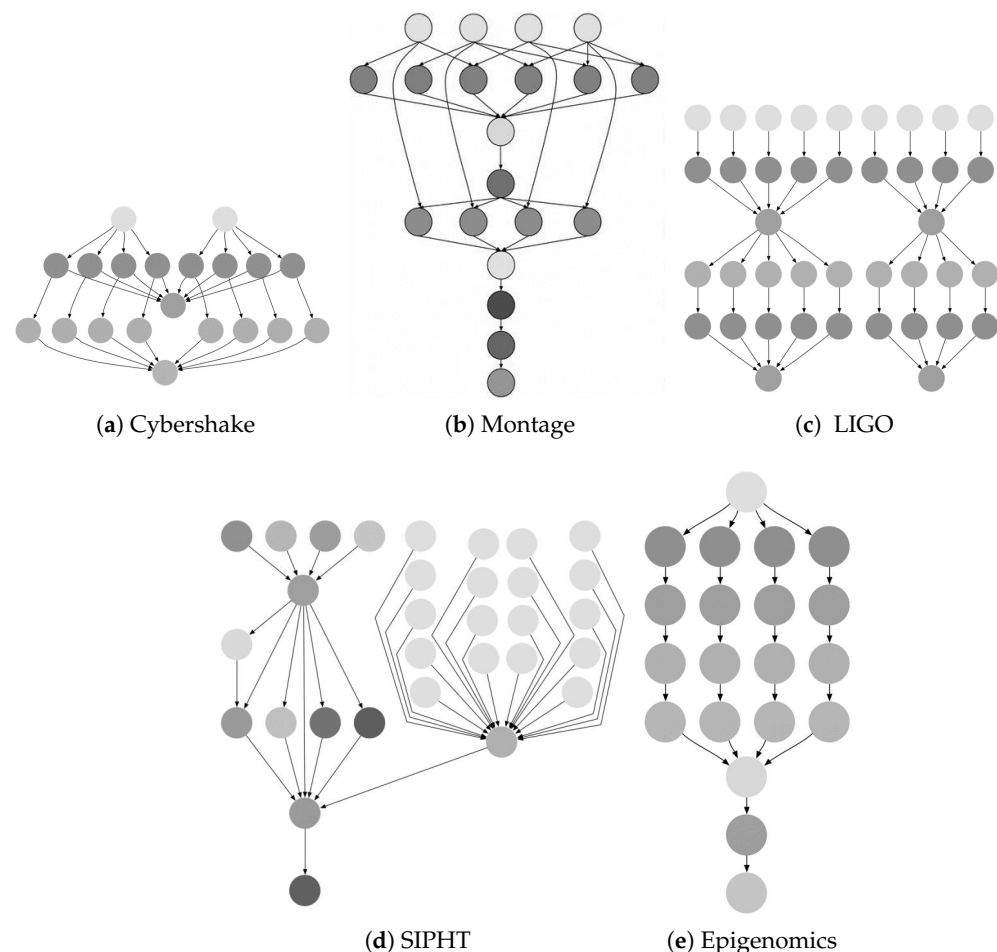


Figure 6. The core structure of workflow applications.

5.3. Experimental Results

For each workflow application, we selected different sizes for scheduling experiments to assess the performance of these algorithms across different scales and complexities, and the outcomes are detailed below:

Based on Table 2, the performance of scheduling algorithms for the Cybershake workflow is evaluated across different task scales (100 and 1000) in terms of makespan and

cost. Among the compared algorithms, HICA achieves the shortest makespan (314.38) for 100 tasks but ranks third for 1000 tasks with a makespan of 1305.88, following Greedy-Ant (1293.68) and ICA (1297.95). In terms of cost, HICA demonstrates outstanding performance, achieving the lowest values for both task scales, at 20,066.17 (100 tasks) and 100,200.61 (1000 tasks), respectively. These results indicate that HICA exhibits significant advantages in cost optimization while also maintaining strong competitiveness in execution time.

Table 2. Experiment results for Cybershake workflow application in different sizes.

Scheduling Algorithm	Makespan (Cybershake)		Cost (Cybershake)	
	100	1000	100	1000
PEFT-GA	322.82	1317.34	20,075.99	100,202.99
GA-PSO	320.52	1303.06	20,072.50	100,200.79
Greedy-Ant	317.65	1293.68	20,091.28	100,207.42
ICA	317.01	1297.95	20,066.67	100,204.14
HICA	314.38	1305.88	20,066.17	100,200.61

Table 3 shows the experimental results for different scheduling algorithms applied to the Montage workflow application at varying task sizes (100 and 1000 tasks), focusing on makespan and cost. For the 100-task case, HICA achieves the shortest makespan (103.13), while PEFT-GA has a relatively higher makespan (104.02). In the 1000-task scenario, HICA also performs well, achieving the lowest makespan (906.89), which was closely followed by Greedy-Ant (911.57). In terms of cost, HICA demonstrates the lowest costs across both task sizes. For instance, for the 1000-task case, HICA's cost is 36,148.09, which is slightly higher than that of GA-PSO at 36,129.15. Overall, HICA showcases strong performance by achieving both low costs and optimized makespan, highlighting its comprehensive advantage.

Table 3. Experiment results for Montage workflow application in different sizes.

Scheduling Algorithm	Makespan (Montage)		Cost (Montage)	
	100	1000	100	1000
PEFT-GA	104.02	914.88	3435.31	36,353.68
GA-PSO	103.93	915.30	3434.86	36,129.15
Greedy-Ant	102.96	911.57	3455.84	36,167.70
ICA	105.05	912.36	3442.04	36,202.52
HICA	103.13	906.89	3434.86	36,148.09

Table 4 presents an analysis of various scheduling algorithms applied to the LIGO workflow at different scales. Among them, the HICA algorithm demonstrates superior performance with the lowest makespan and cost increasing as the workflow size expands from 100 to 1000. Specifically, HICA's makespan rises from 1689.97 to 11,193.08, while its cost increases from 62,989.91 to 686,825.01, showcasing its efficiency in managing larger tasks. In comparison, other algorithms like PEFT-GA, GA-PSO, Greedy-Ant, and ICA show a more pronounced degradation in performance with increased scale, which is indicated by higher makespan and cost values. This analysis highlights HICA's effectiveness in balancing time and cost, making it the most suitable for large-scale LIGO workflow applications among the algorithms tested.

Table 4. Experiment results for LIGO workflow application in different sizes.

Scheduling Algorithm	Makespan (LIGO)		Cost (LIGO)	
	100	1000	100	1000
PEFT-GA	1878.43	11,923.33	63,248.79	687,190.84
GA-PSO	1720.69	11,754.20	62,897.45	686,868.71
Greedy-Ant	1827.25	12,026.83	63,038.26	686,976.39
ICA	1770.51	11,699.45	63,038.64	686,889.78
HICA	1689.97	11,193.08	62,989.91	686,825.01

Table 5 presents the experimental results across different scales (100 and 1000) of SIPHT workflow application, comparing the Makespan and Cost of HICA with those of other algorithms. In terms of Makespan, GA-PSO achieves the shortest execution time for both task scales, with 4471.01 for 100 tasks and 9568.78 for 1000 tasks, while HICA ranks second for 1000 tasks with a Makespan of 9801.55. Regarding cost, HICA demonstrates outstanding performance, achieving the lowest cost for both task scales, at 50,637.07 for 100 tasks and 520,693.86 for 1000 tasks. These results indicate that HICA achieves significant cost optimization while maintaining competitive Makespan performance, and GA-PSO excels in reducing execution time.

Table 5. Experiment results for SIPHT workflow application in different sizes.

Scheduling Algorithm	Makespan (SIPHT)		Cost (SIPHT)	
	100	1000	100	1000
PEFT-GA	4474.77	10,517.32	52,982.04	523,711.12
GA-PSO	4471.01	9568.78	51,491.16	521,853.88
Greedy-Ant	4475.63	10,992.78	52,413.18	522,099.24
ICA	4475.36	10,245.28	51,678.04	521,024.50
HICA	4474.40	9801.55	50,637.07	520,693.86

Table 6 presents the experimental results of five scheduling algorithms (PEFT-GA, GA-PSO, Greedy-Ant, ICA, and HICA) applied to the Epigenomics workflow under two task scales (100 and 997). In terms of Makespan, Greedy-Ant achieves the shortest time for 100 tasks (32,216.08), which is followed closely by HICA (32,286.09). For 997 tasks, HICA demonstrates the best performance with the shortest Makespan (211,805.01). Regarding cost, HICA consistently achieves the lowest values for both task scales with 1,217,159.95 for 100 tasks and 11,611,202.10 for 997 tasks. These results indicate that HICA excels in cost optimization while maintaining strong Makespan performance particularly for larger task scales. Greedy-Ant performs well in Makespan for smaller task scales, while PEFT-GA generally has higher Makespan and cost values compared to the other algorithms.

Table 6. Experiment results for Epigenomics workflow application in different sizes.

Scheduling Algorithm	Makespan (Epigenomics)		Cost (Epigenomics)	
	100	997	100	997
PEFT-GA	34,004.63	207,211.07	1,217,380.43	11,714,322.40
GA-PSO	33,486.41	212,285.94	1,217,308.59	11,627,865.76
Greedy-Ant	32,216.08	215,343.29	1,217,231.84	11,644,505.13
ICA	32,324.28	216,905.62	1,217,202.25	11,680,776.96
HICA	32,286.09	211,805.01	1,217,159.95	11,611,202.10

Based on the results from Figures 7 and 8, we observe significant differences in the performance of various scheduling algorithms (PEFT-GA, GA-PSO, Greedy-Ant, ICA,

and HICA) when applied to five typical scientific workflows (Cybershake, Montage, LIGO, SIPHT, and Epigenomics). The performance, measured by Makespan (execution time), was evaluated for both 100-task and 1000-task scales. For the 100-task scale, the differences in performance between algorithms are relatively small, particularly for the Cybershake and Montage workflows, where the Makespan values are very close across all algorithms. However, HICA demonstrates superior scheduling efficiency in most workflows, achieving the shortest Makespan in both Cybershake and Epigenomics workflows. This highlights the strength of HICA in smaller task scales. In the Epigenomics workflow, Greedy-Ant performs comparably to HICA, while other algorithms, such as PEFT-GA and ICA, fall slightly behind. For LIGO and SIPHT workflows, the performance differences are minor, although GA-PSO and ICA show slightly better results compared to the others. As the task scale increases to 1000 tasks, the performance differences between the algorithms become more pronounced. In the Cybershake workflow, Greedy-Ant achieves the shortest Makespan, outperforming HICA, which drops slightly in its ranking. In the Montage workflow, GA-PSO shows a clear advantage, outperforming all other algorithms. For the LIGO and SIPHT workflows, GA-PSO and Greedy-Ant deliver comparable execution times, while HICA's performance declines slightly compared to the 100-task scale. In contrast, for the Epigenomics workflow, HICA once again demonstrates its strength, achieving the shortest Makespan among all algorithms, highlighting its exceptional capability in handling complex workflows.

makespan results for workflow applications with 100 tasks.

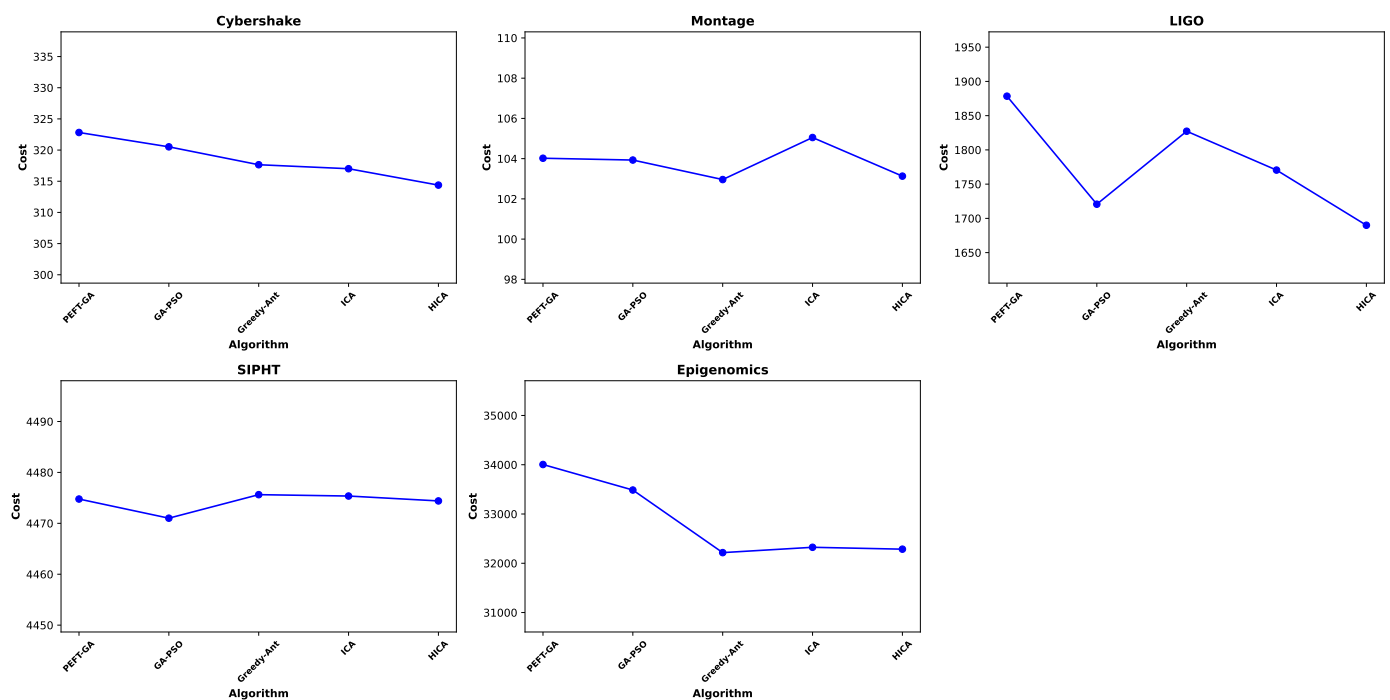


Figure 7. Makespan performance for 100 tasks across scientific workflows. Each subplot represents a different workflow application, and each point in the plot corresponds to the scheduling result of a different algorithm.

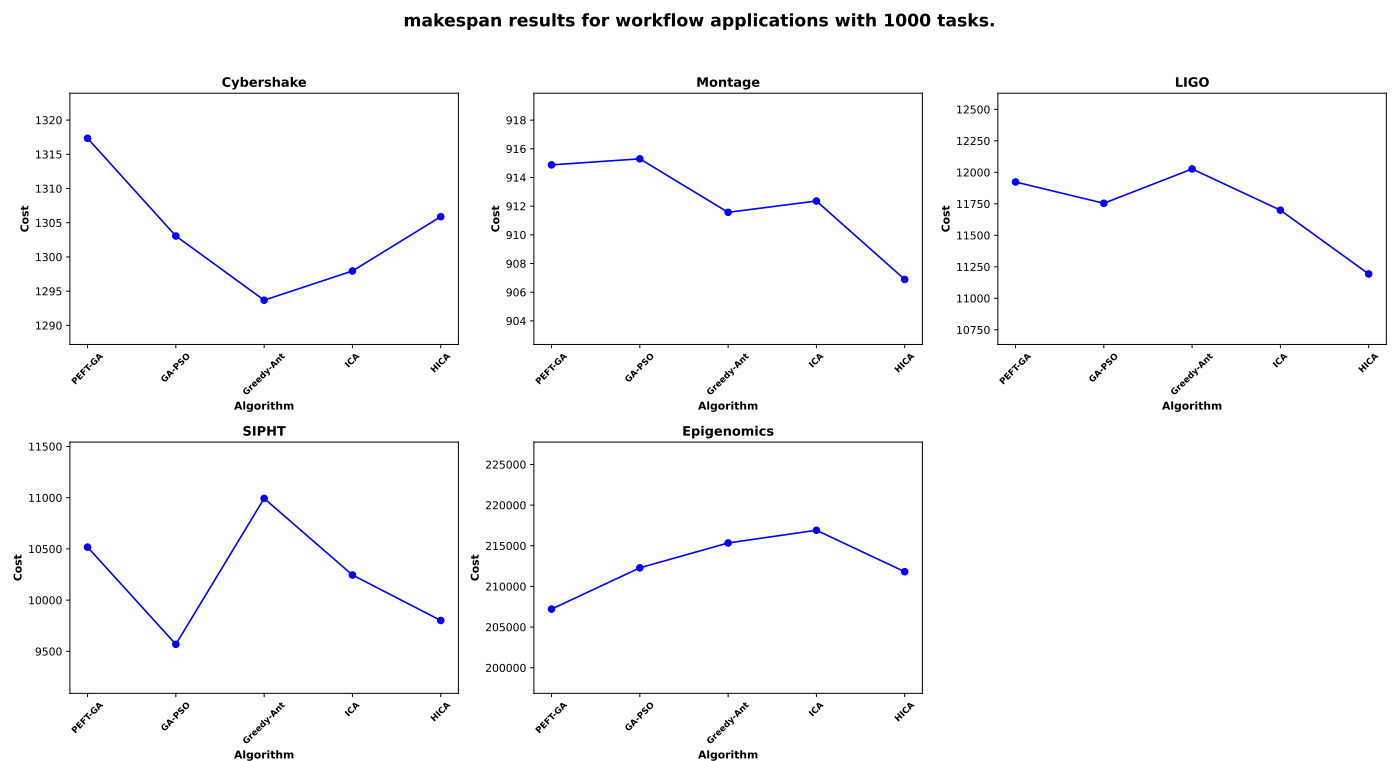


Figure 8. Makespan performance for 1000 tasks across scientific workflows. Each subplot represents a different workflow application, and each point in the plot corresponds to the scheduling result of a different algorithm.

The analysis of Figures 9 and 10 highlights the distinct cost performance of different scheduling algorithms across five scientific workflows applications under task scales of 100 and 1000. At the 100-task scale, HICA consistently achieves the lowest cost across all workflows, demonstrating its strong cost optimization capabilities. Particularly in the Cybershake and Epigenomics workflows, HICA significantly outperforms the other algorithms. Although Greedy-Ant and GA-PSO occasionally show competitive performance in certain workflows, their overall cost-efficiency is still inferior to HICA. PEFT-GA and ICA generally incur higher costs, indicating room for improvement in handling smaller-scale workflows. For the 1000-task scale, HICA continues to lead in cost optimization, securing the lowest costs across all workflows. Its advantage becomes particularly evident in the Epigenomics workflow, where it achieves a remarkably lower cost compared to the other algorithms. This demonstrates HICA's ability to handle large-scale workflows effectively. In contrast, the other algorithms exhibit less consistent performance. PEFT-GA and ICA show relatively high costs, especially in complex workflows such as Epigenomics and SIPHT, suggesting limitations in their scalability. While Greedy-Ant and GA-PSO improve in some cases, they are still outperformed by HICA in all scenarios. These findings underline HICA's superiority in cost optimization and scalability across different task scales. Its consistent performance, regardless of workflow type or scale, makes it a highly reliable choice for cost-sensitive scientific workflow applications. In comparison, the other algorithms may be suitable for specific workflows or scales but lack the overall robustness and adaptability of HICA. This emphasizes the importance of choosing an algorithm that balances cost, scalability, and adaptability, with HICA emerging as the optimal choice for diverse and large-scale workflow scenarios.

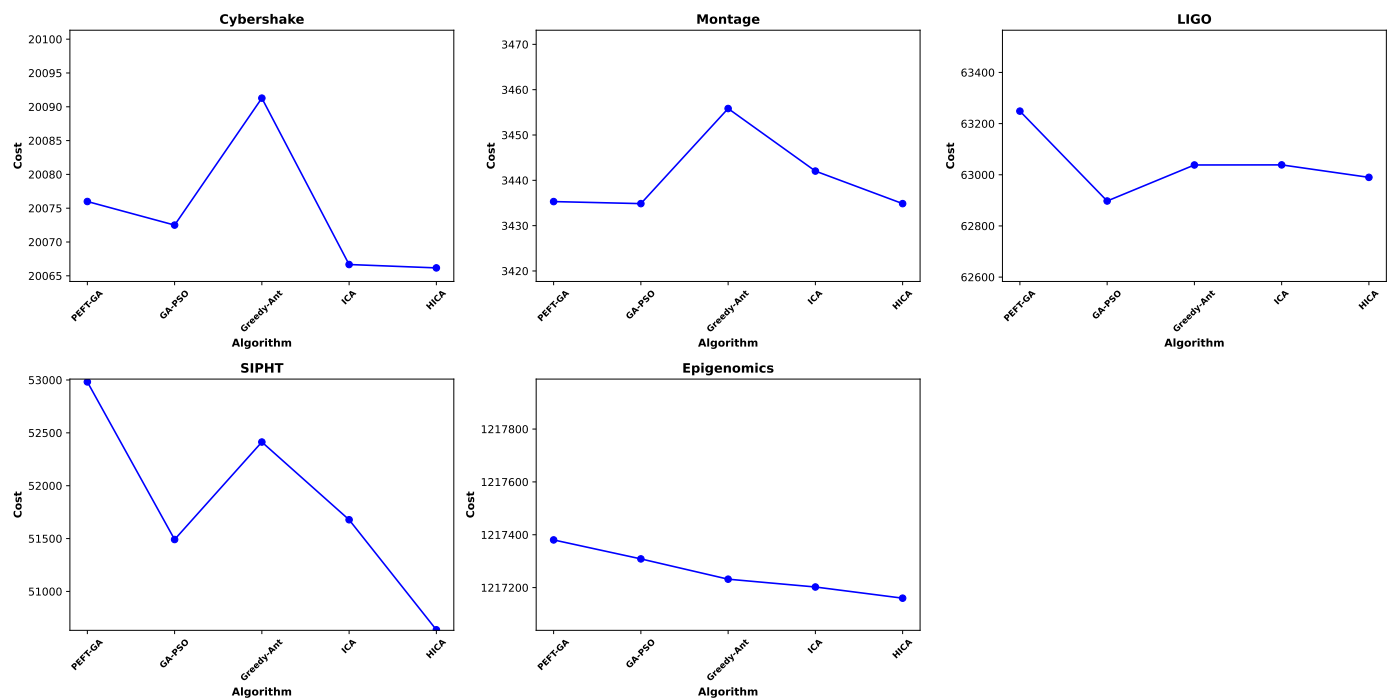
cost results for workflow applications with 100 tasks.

Figure 9. Cost performance for 100 tasks across scientific workflows. Each subplot represents a different workflow application, and each point in the plot corresponds to the scheduling result of a different algorithm.

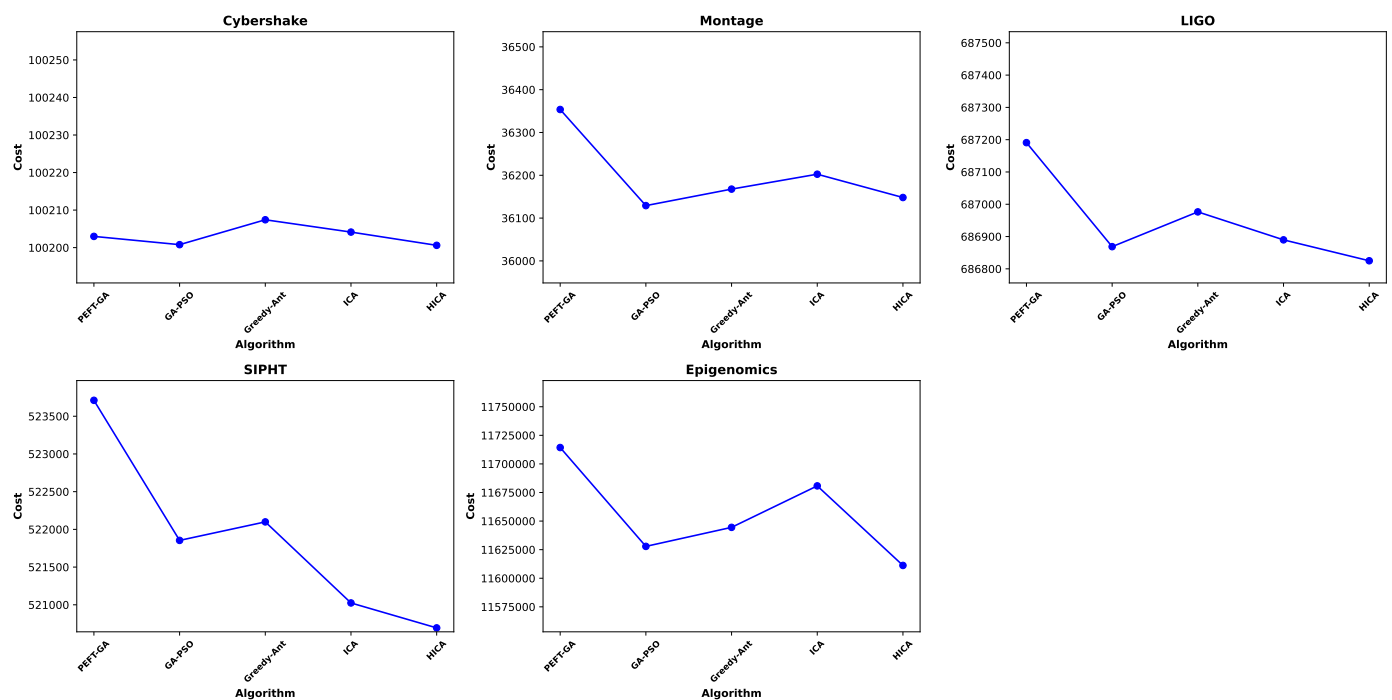
cost results for workflow applications with 1000 tasks.

Figure 10. Cost performance for 1000 tasks across scientific workflows. Each subplot represents a different workflow application, and each point in the plot corresponds to the scheduling result of a different algorithm.

5.4. An Actual Application Scenario of HICA: Earth System Model (ESM) Parameter Tuning

To further test the scheduling performance of HICA, we added real-world scientific workflow application scenarios based on the previous section. We selected a scientific workflow for tuning the parameters of Earth system models in atmospheric science, which is widely used in climate research. Running Earth system models is a typical data-intensive process, with global climate simulations potentially lasting decades or even centuries, generating TB- to PB-level data. This not only leads to a significant consumption of storage resources but also requires the parallel involvement of numerous computational nodes to accelerate the execution of complex computations. For scientific research applications of this scale, introducing workflows can effectively manage both computational and storage resources while modularizing and formalizing complex scientific computing processes, thereby improving overall efficiency.

A complete Earth system model parameter tuning workflow consists of the following parts:

Preprocessing: This involves the collection, cleaning, and preparation of input data, such as climate variables and initial model parameters, for model simulations.

Model Simulation: The Earth system model (ESM) is executed using the prepared input data. This step may involve multiple iterations to simulate various climate scenarios over extended time periods, often spanning decades or centuries.

Parameter Tuning: This step adjusts the model parameters to optimize the accuracy of the simulation results. Various optimization techniques, including surrogate models, genetic algorithms, may be used to find the best parameter settings.

Postprocessing: After the simulations, the results are analyzed and compared against observational data or benchmarks to evaluate the performance of the model. This may include error analysis, visualization of results, and statistical testing.

We apply the HICA algorithm proposed in this paper to parameter tuning in Earth System Models, evaluating the changes in workflow makespan and cost after introducing the scheduling algorithm. We normalize the results by using the makespan and cost in the initial scenario without any scheduling algorithm as the baseline and compare the changes after applying HICA. The results are shown in Figure 11. The HICA algorithm achieves varying degrees of improvement in both makespan and cost in workflow execution with a 13% improvement in makespan and a more significant 21% improvement in cost. These results demonstrate that the HICA algorithm not only effectively addresses the scientific workflow scheduling issues encountered in WorkflowSim but also significantly enhances the execution efficiency of workflows in specific scientific application scenarios.

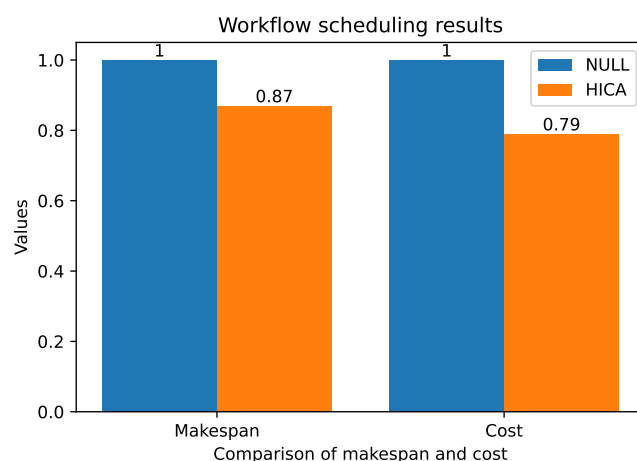


Figure 11. Scheduling results for the ESM parameter tuning workflow, NULL represents the original scenario without using any algorithm.

6. Conclusions

In this paper, we propose a hybrid scientific workflow scheduling algorithm, HICA, aimed at optimizing the execution time and cost of scientific workflows in cloud environments. The Imperial Competitive Algorithm (ICA) has excellent optimization capabilities, enabling it to escape local optima through operations such as colony assimilation and revolution. HEFT, on the other hand, is an efficient list-based scheduling algorithm that can quickly find a scheduling sequence that meets the requirements. Our proposed HICA takes full advantage of the strengths of both algorithms by integrating HEFT as one of the initial solutions into the initial population of the ICA. By improving the quality of the initial solution in the ICA, the algorithm reduces iterations in low-quality regions, allowing it to find a high-quality scheduling sequence more quickly.

To validate the effectiveness of the HICA algorithm, we conducted scheduling experiments using five real-world scientific workflow applications: Cybershake, Montage, LIGO, SIPHT, and Epigenomics, with different workflow scales. The experimental results show that compared to other hybrid scheduling algorithms, the HICA algorithm demonstrates superior scheduling performance across workflows of varying node scales with particularly outstanding performance in large-scale workflow scheduling experiments.

In future work, we plan to explore scheduling methods targeting additional objectives, such as energy consumption-based scheduling. Furthermore, we aim to investigate new optimization algorithms and explore how to integrate these methods into the scheduling of scientific workflows. Finally, scheduling workflows in heterogeneous cloud environments across multiple data centers will also be a key direction for future research.

Author Contributions: Conceptualization, L.H.; methodology, X.W.; validation, X.C.; writing—original draft preparation, L.H., X.W. and X.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R&D Plan of China under Grant No. 2017YFA0604500, Key scientific and technological R&D Plan of Jilin Province of China under Grant No. 20180201103GX, the National Key Research and Development Plan of China under Grant No. 2020YFB0204800, the National Natural Science Foundation of China under Grant No. T2125006, and the Jiangsu Innovation Capacity Building Program under Grant No. BM2022028.

Data Availability Statement: The data used and analyzed during the current study are available from the corresponding author on reasonable request.

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

Appendix A

Table A1. State of the art related to hybrid scientific workflow scheduling algorithm.

Performance Metrics	Method	References
Deadline and budget	GA	[13]
Deadline	ACO	[14]
Deadline and cost	PSO	[15]
Cost	CSO	[16]
Cost and load balance	BA	[17]
Makespan, reliability and energy consumption	NSGA-II	[18]
Deadline and budget	PSO, BDHEFT	[23]
Makespan and cost	ALO, SCA	[24]
Deadline and cost	EAFSA, IC-PCP and EAFSAIPR with HECC	[25]

Table A1. Cont.

Performance Metrics	Method	References
Makespan and cost	GA-HEFT	[26]
Makespan, cost, energy, and throughput	SOA, GOA	[27]
Makespan, cost and load	ALO, PSO	[28]
SLR, speed up and efficiency	HDPSO	[29]
Makespan and resource utilization	HEFT, PSO and GA	[30]
Makespan and cost	GSA, HEFT	[31]
Makespan and cost	CS, FPA	[32]

References

1. Talia, D. Workflow systems for science: Concepts and tools. *Int. Sch. Res. Not.* **2013**, *2013*, 404525. [\[CrossRef\]](#)
2. Vaquero, L.M.; Roderio-Merino, L.; Caceres, J.; Lindner, M. A break in the clouds: Towards a cloud definition. *ACM SIGCOMM Comput. Commun. Rev.* **2008**, *39*, 50–55. [\[CrossRef\]](#)
3. Gong, C.; Liu, J.; Zhang, Q.; Chen, H.; Gong, Z. The characteristics of cloud computing. In Proceedings of the 2010 39th International Conference on Parallel Processing Workshops, San Diego, CA, USA, 13–16 September 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 275–279.
4. Arabnejad, H.; Barbosa, J.G. List scheduling algorithm for heterogeneous systems by an optimistic cost table. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 682–694. [\[CrossRef\]](#)
5. Arabnejad, H.; Barbosa, J.G.; Prodan, R. Low-time complexity budget–deadline constrained workflow scheduling on heterogeneous resources. *Future Gener. Comput. Syst.* **2016**, *55*, 29–40. [\[CrossRef\]](#)
6. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
7. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [\[CrossRef\]](#)
8. Deelman, E.; Vahi, K.; Juve, G.; Rynge, M.; Callaghan, S.; Maechling, P.J.; Mayani, R.; Chen, W.; Da Silva, R.F.; Livny, M.; et al. Pegasus, a workflow management system for science automation. *Future Gener. Comput. Syst.* **2015**, *46*, 17–35. [\[CrossRef\]](#)
9. Ludäscher, B.; Altintas, I.; Berkley, C.; Higgins, D.; Jaeger, E.; Jones, M.; Lee, E.A.; Tao, J.; Zhao, Y. Scientific workflow management and the Kepler system. *Concurr. Comput. Pract. Exp.* **2006**, *18*, 1039–1065. [\[CrossRef\]](#)
10. Hull, D.; Wolstencroft, K.; Stevens, R.; Goble, C.; Pocock, M.R.; Li, P.; Oinn, T. Taverna: A tool for building and running workflows of services. *Nucleic Acids Res.* **2006**, *34*, W729–W732. [\[CrossRef\]](#)
11. Fahringer, T.; Prodan, R.; Duan, R.; Hofer, J.; Nadeem, F.; Nerieri, F.; Podlipnig, S.; Qin, J.; Siddiqui, M.; Truong, H.L.; et al. Askalon: A development and grid computing environment for scientific workflows. In *Workflows for e-Science: Scientific Workflows for Grids*; Springer: London, UK, 2007; pp. 450–471.
12. Konjaang, J.K.; Xu, L. Meta-heuristic approaches for effective scheduling in infrastructure as a service cloud: A systematic review. *J. Netw. Syst. Manag.* **2021**, *29*, 15. [\[CrossRef\]](#)
13. Yu, J.; Buyya, R. Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Sci. Program.* **2006**, *14*, 217–230. [\[CrossRef\]](#)
14. Chen, W.N.; Zhang, J.; Yu, Y. Workflow scheduling in grids: An ant colony optimization approach. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 3308–3315.
15. Rodriguez, M.A.; Buyya, R. Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. *IEEE Trans. Cloud Comput.* **2014**, *2*, 222–235. [\[CrossRef\]](#)
16. Bilgaiyan, S.; Sagnika, S.; Das, M. Workflow scheduling in cloud computing environment using Cat Swarm Optimization. In Proceedings of the Souvenir of the 2014 IEEE International Advance Computing Conference, IACC 2014, New Delhi, India, 21–22 February 2014; pp. 680–685.
17. Sagnika, S.; Bilgaiyan, S.; Mishra, B.S.P. Workflow scheduling in cloud computing environment using bat algorithm. In Proceedings of the First International Conference on Smart System, Innovations and Computing: SSIC 2017, Jaipur, India, 14–16 April 2017; Springer: Berlin/Heidelberg, Germany, 2018; pp. 149–163.
18. Rehani, N.; Garg, R. Meta-heuristic based reliable and green workflow scheduling in cloud computing. *Int. J. Syst. Assur. Eng. Manag.* **2018**, *9*, 811–820. [\[CrossRef\]](#)
19. Połap, D.; Woźniak, M. Red fox optimization algorithm. *Expert Syst. Appl.* **2021**, *166*, 114107. [\[CrossRef\]](#)
20. Zervoudakis, K.; Tsafarakis, S. A global optimizer inspired from the survival strategies of flying foxes. *Eng. Comput.* **2023**, *39*, 1583–1616. [\[CrossRef\]](#)
21. Połap, D.; Woźniak, M. Polar bear optimization algorithm: Meta-heuristic with fast population movement and dynamic birth and death mechanism. *Symmetry* **2017**, *9*, 203. [\[CrossRef\]](#)

22. Zervoudakis, K.; Tsafarakis, S. A mayfly optimization algorithm. *Comput. Ind. Eng.* **2020**, *145*, 106559. [\[CrossRef\]](#)
23. Verma, A.; Kaushal, S. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. *Parallel Comput.* **2017**, *62*, 1–19. [\[CrossRef\]](#)
24. 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. [\[CrossRef\]](#)
25. Ramathilagam, A.; Vijayalakshmi, K. Workflow scheduling in cloud environment using a novel metaheuristic optimization algorithm. *Int. J. Commun. Syst.* **2021**, *34*, e4746. [\[CrossRef\]](#)
26. Aziza, H.; Krichen, S. A hybrid genetic algorithm for scientific workflow scheduling in cloud environment. *Neural Comput. Appl.* **2020**, *32*, 15263–15278. [\[CrossRef\]](#)
27. Mohammadzadeh, A.; Masdari, M. Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm. *J. Ambient Intell. Humaniz. Comput.* **2023**, *14*, 3509–3529. [\[CrossRef\]](#)
28. Kakkottakath Valappil Thekkepurayil, J.; Suseelan, D.P.; Keerikkattil, P.M. An effective meta-heuristic based multi-objective hybrid optimization method for workflow scheduling in cloud computing environment. *Clust. Comput.* **2021**, *24*, 2367–2384. [\[CrossRef\]](#)
29. Shirvani, M.H. A hybrid meta-heuristic algorithm for scientific workflow scheduling in heterogeneous distributed computing systems. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103501. [\[CrossRef\]](#)
30. Mikram, H.; El Kafhali, S.; Saadi, Y. HEPGA: A new effective hybrid algorithm for scientific workflow scheduling in cloud computing environment. *Simul. Model. Pract. Theory* **2024**, *130*, 102864. [\[CrossRef\]](#)
31. 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. [\[CrossRef\]](#)
32. Sharma, G.; Khurana, S.; Harnal, S.; Lone, S.A. CSFPA: An intelligent hybrid workflow scheduling algorithm based upon global and local optimization approach in cloud. *Concurr. Comput. Pract. Exp.* **2022**, *34*, e7176. [\[CrossRef\]](#)
33. Chen, W.; Deelman, E. Workflowsim: A toolkit for simulating scientific workflows in distributed environments. In Proceedings of the 2012 IEEE 8th International Conference on E-Science, Chicago, IL, USA, 8–12 October 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 1–8.
34. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.; Buyya, R. CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exp.* **2011**, *41*, 23–50. [\[CrossRef\]](#)
35. Nagar, R.; Gupta, D.K.; Singh, R.M. Time effective workflow scheduling using genetic algorithm in cloud computing. *Int. J. Inf. Technol. Comput. Sci.* **2018**, *10*, 68–75. [\[CrossRef\]](#)
36. Manasrah, A.M.; Ba Ali, H. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1934784. [\[CrossRef\]](#)
37. Xiang, B.; Zhang, B.; Zhang, L. Greedy-ant: Ant colony system-inspired workflow scheduling for heterogeneous computing. *IEEE Access* **2017**, *5*, 11404–11412. [\[CrossRef\]](#)
38. Arshad, R.; Rafeh, R. Deadline-constrained workflow scheduling using imperialist competitive algorithm on infrastructure as a service clouds. In Proceedings of the 2015 2nd International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, Iran, 5–6 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 835–842.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.