electronics

Article

An Enhanced PSO Algorithm for Scheduling Workflow Tasks in Cloud Computing

Samar Hussni Anbarkhan [1,*] and Mohamed Ali Rakrouki [2,3,4]

[1] Information Systems Department, Northern Border University, Arar 73213, Saudi Arabia
[2] College of Computer Science and Engineering, Taibah University, Medina 42353, Saudi Arabia; mrakrouki@taibahu.edu.sa
[3] Ecole Supérieure des Sciences Economiques et Commerciales de Tunis, University of Tunis, Tunis 1089, Tunisia
[4] Business Analytics and DEcision Making Lab (BADEM) at Tunis Business School, University of Tunis, Bir El Kassaa 2059, Tunisia
* Correspondence: samar.hussni@nbu.edu.sa

Abstract: This paper proposes an enhanced Particle Swarm Optimization (PSO) algorithm in order to deal with the issue that the time and cost of the PSO algorithm is quite high when scheduling workflow tasks in a cloud computing environment. To reduce particle dimensions and ensure initial particle quality, intensive tasks are combined when scheduling workflow tasks. Next, the particle initialization is optimized to ensure better initial particle quality and reduced search space. Then, a suitable self-adaptive function is integrated to determine the best direction of the particles. The experiments show that the proposed enhanced PSO algorithm has better convergence speed and better performance in the execution of workflow tasks.

Keywords: task scheduling; cloud computing; metaheuristics; particle swarm optimization

Citation: Anbarkhan, S.H.; Rakrouki, M.A. An Enhanced PSO Algorithm for Scheduling Workflow Tasks in Cloud Computing. *Electronics* **2023**, *12*, 2580. https://doi.org/10.3390/electronics12122580

Academic Editor: Ying Tan

Received: 2 May 2023
Revised: 2 June 2023
Accepted: 5 June 2023
Published: 7 June 2023

1. Introduction

Due to its high performance and distributed computing capabilities, cloud computing is widely used. The benefits of cloud computing include virtualization, high scalability, high dependability, on-demand service, huge size, and low cost [1]. As a result, more and more researchers are beginning to focus on this field. There has been a boom in research on resource usage, storage performance, and cloud computing system performance. According to their characteristics, task scheduling assigns the users' requested tasks to the appropriate cloud resources for execution. The dependability, availability, and resource usage of cloud computing are most significantly impacted by the effectiveness of job scheduling. How to equitably assign jobs of various types and requirements to suitable computer resources is the focus and challenge of task scheduling algorithm research in cloud computing [2].

Because of its commercial nature, cloud computing is leased to users as a service provision model in distributed computing. Users use the services provided by cloud computing, but they do not understand how tasks are processed inside cloud computing networks. Cloud computing uses its powerful distributed processing capabilities to handle huge amounts of tasks. Traditional task scheduling is mainly divided into independent task scheduling and workflow task scheduling. Independent task scheduling has no order relationship because the tasks are relatively independent and there is no priority in the execution of tasks. Therefore, the scheduling of independent tasks is relatively simple. This type of scheduling cannot be used to describe most large-scale applications. Task scheduling in the cloud computing environment mainly refers to workflow task scheduling.

As shown in Figure 1, task scheduling in a cloud computing environment usually divides a large task set into several tasks, and then assigns these tasks to appropriate computing resources for processing according to the specific requirements of users, and returns

the processing results to users. The process of task scheduling is relatively complicated. At the beginning of scheduling,the task set is split and decomposed, divided into several tasks, and computing resources are allocated to the tasks. This is the task scheduling stage; after the computing resources are assigned a task, they execute the task, obtain the execution result of the task, and return the execution result to the user. This is the task execution phase. In the process of task execution, there are two situations. One is that there is no sequence relationship between tasks, and the execution of tasks can be processed in parallel. This kind of task is mainly an independent task; the other is that data transmission is required between tasks. There are dependencies, and tasks need to be executed sequentially. This kind of task is mainly a workflow task. No matter what kind of tasks are performed, the main goal is to meet the needs of users. This is determined by the service-oriented business model of cloud computing. It is required to ensure that tasks are error-free and smoothly executed while meeting user quality requirements.

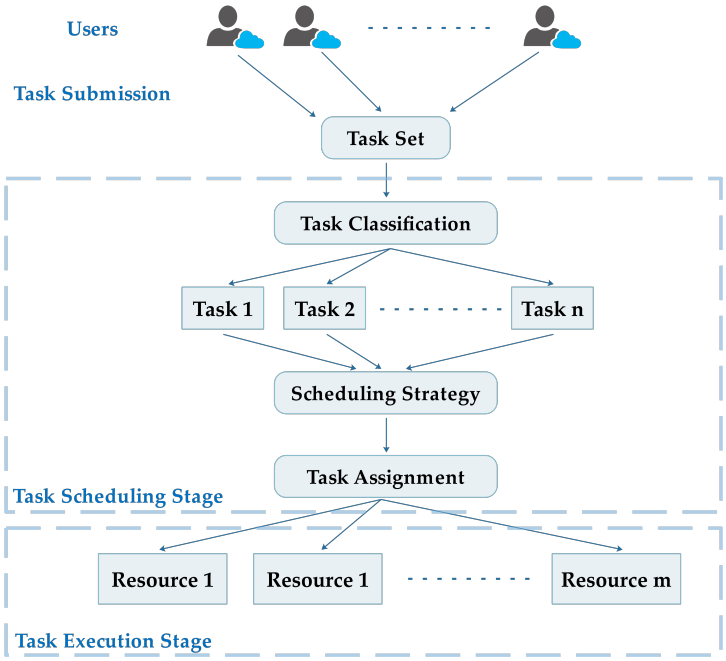

Figure 1. Task Scheduling.

Therefore, the core of task scheduling is allocating the proper resources to jobs in accordance with user needs so that it can satisfy quality of service (QoS) standards such as time and cost. Cloud computing is well-liked by users because of its powerful computational capacity and low cost. It draws an increasing number of researchers that are interested in studying its distributed computing, cloud storage, and virtualization technologies. Task scheduling for cloud computing is increasingly carried out using Particle Swarm Optimization (PSO), as an effective heuristic technique.

Task scheduling in cloud computing is an $\mathcal{NP}$-hard problem [3]. The PSO algorithm is widely used in solving this problem due to its high efficiency and fast convergence speed [4]. A hybrid PSO algorithm that incorporates genetic algorithm operators (i.e., crossover and mutation) was proposed by Xue and Wu [5]. According to experimental findings, this approach outperforms the traditional PSO algorithm in terms of cost minimization for a given execution time. To reduce the overall task flow's execution costs, Guo et al. [6] suggested a task scheduling model and a PSO technique based on tiny position value criteria. The experimental results demonstrate that the original algorithm has a rapid convergence speed and is faster than the other two algorithms when compared to the one incorporated with crossover and mutation. The research of Varalakshmi et al. [7] on the PSO algorithm aims to meet the user's QoS requirements in workflow scheduling and can effectively improve CPU utilization. A new hybrid PSO algorithm (HPSO) based on heuristic task

scheduling that takes into account the computation cost and data transmission cost was proposed by Pandey et al. [8]. By lowering the cost of calculation and communication, it is applied to workflow applications. The experimental findings demonstrate that the HPSO algorithm can reduce costs and allocate sensible resources when scheduling tasks. The impact of cutting the time cost is still being determined. The user's QoS requirements are the subject of the aforementioned study and PSO algorithm improvement, which focuses on time, cost, and CPU utilization optimization. The PSO method is used to address the issue of independent job scheduling in the cloud computing system. Excellent performance, although the outcome could have been more pleasing for the workflow job scheduling. The industry rarely engages in research concerning the application of the PSO algorithm in workflow task scheduling scenarios in a cloud computing environment.

After an in-depth investigation of previous research results, this paper proposes an Enhanced PSO (EPSO) algorithm based on establishing a workflow task model, which improves the initialization operation and adaptive function of the particles to meet the user's time, cost, and CPU utilization requirements. The workflow task model processing is designed, the particle dimensions are reduced, the particle swarm optimization process is optimized, and the execution completion time of the workflow task scheduling in the cloud computing environment is minimized.

2. Related Works

2.1. Problem Formulation

In a cloud computing environment, users submit multiple tasks to the cloud platform for scheduling and processing. The tasks submitted by the user to the cloud platform will enter the scheduling list and be distributed to the virtual machines in the cloud platform for execution through the scheduler.

When the user has a service demand, they send a request to the cloud system to request the cloud system to provide services, and the cloud system will dispatch the resources of the data center to the user. For cloud service providers, the quality of task scheduling directly affects their income. There are two types of task scheduling in cloud computing: independent task scheduling and workflow task scheduling. The dependencies between tasks do not need to be considered in independent task scheduling and there is no sequence of task execution; workflow task scheduling needs to consider the dependencies and data transmission between tasks, and task execution has a sequence. The task scheduling algorithm in cloud computing needs to consider different types of task scheduling effects, reasonably map different tasks to appropriate resources for execution, and ensure service quality, such as the shortest execution time and the smallest execution cost.

Definition 1. *The set of resources in a cloud computing system is expressed as* $R = (r_1, r_2, ..., r_m)$*, where m represents the total number of computing resources. Each resource is represented as follows:*

$$r_j = (r_{id}^j, r_{cap}^j) \tag{1}$$

where r_{id} *represents the ID of the computing resource and* r_{cap} *represents the performance attribute of the resource. There are many performance attributes of resources. This section mainly considers the attributes of resources in terms of CPU performance, communication bandwidth, charging price, etc., to facilitate the subsequent calculation of task execution time, data transmission time, and task execution cost. Therefore,* r_{cap} *is expressed as follows:*

$$r_{cap} = (r_{comp}, r_{bw}, r_{price}) \tag{2}$$

where r_{comp} *is the CPU performance of the computing resource, which measures the computing power of the resource;* r_{bw} *is the communication bandwidth of the computing resource, which measures the data transmission capability of the resource; and* r_{price} *represents the charging price of the computing resource.*

Definition 2. *A directed acyclic graph $G = (T, E, \omega)$ represents the task model in the cloud, where $T = \{t_1, t_2, ..., t_n\}$ represents the task set, E represents the dependencies between tasks (that is, $(T_i, T_j) \in E$), and $\omega : E \to \mathbb{R}$ represents the weight function of G, representing the amount of data that needs to be transmitted between tasks, where n represents the total number of tasks. The following four-tuple represents the attribute characteristics of task t_i, namely,*

$$t_i = (t^i_{id}, t^i_{length}, t^i_{rres}) \tag{3}$$

where t_{id} represents the label information of the task; t_{length} represents the amount of data that needs to be calculated when the task is executed; and t_{rres} represents the requirement of the task execution for the allocated resource attributes.

The attributes of resources are mainly manifested in aspects such as CPU performance, communication bandwidth, and computing price. Therefore, the task's requirements for computing resource attributes are also mainly reflected in the following four aspects:

$$t_{rres} = (t_{comp}, t_{bw}, t_{price}) \tag{4}$$

The parameter t_{comp} of t_{rres} indicates the minimum requirement of CPU performance for this task; t_{bw} indicates the minimum requirement of communication bandwidth for this task; and t_{price} indicates the maximum limit of usage price for this task.

The execution completion time of task i is determined by the amount of data required to execute the task and the CPU performance of the computing resources. When the attribute characteristics of computing resource r_j meet the computing resources requirements of task t_i , the algorithm assigns task t_i to computing resource r_j for execution. The execution time $ET(i, j)$ of task t_i on resource r_j is calculated as follows:

$$ET(i, j) = \frac{t^i_{length}}{r^j_{comp}} \tag{5}$$

The approximate execution cost of task t_i on computing resource r_j is calculated as follows:

$$EC(i, j) = ET(i, j) * r^j_{price} \tag{6}$$

Definition 3. *Matrix $W = T \times R$ represents the task execution time cost matrix, where w_{ij} is calculated according to Equation (5), representing the time spent by task t_i on resource r_j:*

Definition 4. *Matrix $C = R \times R$ measures the communication capability between resource r_j and resource c_j, that is, the data transmission capability.*

Definition 5. *The task resource allocation strategy is defined as matrix X. For a given task set T and resource set R, the allocation strategy description is shown in Equation (7):*

$$X = \{x_{11}, x_{12}, ..., x_{1m}, x_{21}, x_{22}, ..., x_{nm}\} \tag{7}$$

where x_{ij} takes a value of 0 or 1, indicating whether to assign task t_i to resource r_j or not.

Let us define task t_i to execute at start time point $St(i, j)$ and execution completion time point $Ft(i, j)$ on resource r_j as follows:

$$St(i, j) = \max_{t_k \in pred(t_i)} \{Ft(k, r)\} \tag{8}$$

$$Ft(i, j) = St(i, j) + w_{ij} \tag{9}$$

Then, the maximum completion time of all tasks is as follows:

$$makespan = \max\{Ft(i, j), 1 \le i \le n, 1 \le j \le m\} \tag{10}$$

Task t_i is executed on resource r_j. The start time point $St(i,j)$ is equal to the maximum completion time of all predecessor tasks of task t_i; task t_i is executed on resource r_j. The completion time $Ft(i,j)$ when task t_i finishes executing on resource r_j is equal to the sum of the start time point when task t_i starts executing on resource r_j and the execution time of task t_i on resource r_j.

How to optimize the PSO algorithm to adapt to the workflow task scheduling in the cloud computing environment, how to find an optimal task resource allocation strategy X, and how to minimize the completion time of the workflow task execution are the main problems to be solved in this paper. According to this description, the objective function is minimizing the *makespan*.

2.2. Literature Review

One of the most notable problems with cloud computing systems, which are made up of numerous heterogeneous computing resources, is the task scheduling problem [9–13].

In the literature, among many metaheuristics, the PSO algorithm has been used to optimize task scheduling in a cloud computing environment. A small position-value-based PSO algorithm was introduced by Guo et al. [6] to address the issue of task scheduling and transfer costs in cloud computing. Studies have shown that when the resource search range is vast, this algorithm performs better and converges faster than the conventional PSO algorithm. Awad et al. [14] considered reliability and availability when scheduling tasks and proposed an enhanced PSO algorithm using a mathematical model. Mirzayi and Rafe [15] proposed a hybrid PSO algorithm based on a gravitation search algorithm. In order to optimize the QoS in a cloud computing environment, Xue et al. [16] proposed a dynamic-adapting PSO algorithm based on a price model. In comparison to the PSO, gravitational search, and genetic-gravitational algorithms, the suggested technique has demonstrated good performance. Huang et al. [17] proposed several PSO variants using a logarithm-decreasing strategy to generate a better scheduling scheme. The experimental results have shown promising results compared to some population-based approaches. Alsaidy et al. [18] proposed a PSO algorithm by improving the initialization stage using two list-based heuristics. The experiments have proven the superiority of the improved procedure compared to the traditional PSO algorithm.

The literature has taken into account various metaheuristics for dealing with task scheduling in cloud platforms. Su et al. [19] suggested an improved Ant Colony Optimization (ACO) algorithm for scheduling tasks to minimize the time and cost of task completion, reduce migration time, and increase user satisfaction. Hamed and Alkinani [20] presented a genetic algorithm (GA) to minimize task completion times and execution costs while maximizing resource utilization. Chaudhary et al. [21] considered task scheduling to satisfy QoS requirements using a Harmony Search algorithm. For the same objective, Gabi et al. [22] proposed a hybrid approach by combining Simulated Annealing and Cat Swarm optimization algorithms. Chen and Long [23] proposed a hybrid PSO and ACO algorithm. The experimental results showed better task scheduling performance regarding cost and running time. Jia et al. [24] considered the problem of task scheduling in a cloud computing environment with the objective of optimizing execution time, consumption, and virtual machine load. To deal with this problem, the authors proposed an improved whale optimization algorithm using an inertial weight-based local search strategy.

Various recent literature reviews on task scheduling in cloud computing environments can be found in [11–13,25,26].

It is worth noting that PSO is an excellent option for optimizing a wide range of real-world optimization problems and applications. Recently, Shami et al. [27] presented a thorough analysis of PSO, covering its fundamental ideas, binary PSO, neighborhood structures, existing and historical versions, outstanding engineering uses, and its shortcomings.

2.3. Particle Swarm Optimization Algorithm

In this section, our proposed PSO algorithm is described. First, a detailed description of the classical PSO algorithm is provided. After that, the different techniques used to improve it are provided, as well as a pseud-code of the algorithm's implementation.

2.3.1. Principle of the PSO Algorithm

A PSO algorithm is an evolutionary search algorithm proposed by Kennedy [4], which is mainly inspired by the behavior of birds and animals looking for food. The PSO algorithm is well-liked because it lacks evolutionary processes such as crossover and mutation. It is more straightforward than other evolutionary algorithms, such as the genetic algorithm (GA).

A PSO algorithm regards each particle as a feasible solution to the problem. These feasible solutions correspond to the collection of resources in the cloud computing system. Every particle in the search space is a position vector, and the number of dimensions it has corresponds to the number of tasks. Resources in the cloud are represented by the number of particles. At the beginning of the algorithm, a group of particles is randomly initialized, that is, each particle is given a random initial position and initial velocity. In other literature, the description of speed is different. This paper adopts the description formula of Kennedy [4] for speed as follows:

$$v_i^{k+1} = v_i^k + \varphi_1 rd_1(p_{besti}^k - x_i^k) + \varphi_2 rd_2(g_{best} - x_i^k) \tag{11}$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \tag{12}$$

where k is the number of iterations, rd_1 and rd_2 are random numbers, and φ_1 and φ_2 are learning factors. x_i^k and x_i^{k+1} are the current and the next step positions of the particle i, respectively. v_i^k and v_i^{k+1} are the current speed and the next step speed of the particle i, respectively. p_{besti}^k is the best position experienced by the particle i, that is, the best historical position. g_{best} is the best position experienced by all particles, that is, the global best position.

The best historical position is the best position obtained by the particle so far, and the global best position is the best value of all the local best positions obtained so far. In each iteration, the particle updates its position and velocity information by using the historical or global best positions.

The speed formula has three essential parts. The first part v_i^k represents the past velocity. If no external force is applied to it, the particle will continue to travel in the original direction, which is called the law of inertia. The second part $\varphi_1 rd_1(p_{besti}^k - x_i^k)$ represents the individual characteristics of a particle, that is, the particle's previous motion experience. The individual characteristics of particles mainly affect the local search ability of particles. The third part $\varphi_2 rd_2(g_{best} - x_i^k)$ represents the characteristics of the group, that is, the influence of the position information g_{best} of other particles on the speed and position of the current particle mainly affects the global search ability of the particle. In the case of only the first part, the particles keep moving at the past speed when they do not reach the boundary value, so the probability of reaching the optimal solution is minimal. In the case of only the first and second parts, the particle shows a local search ability. The particle's search space gradually shrinks over the iterative process, making it likely to settle on a local optimal solution.

Shi and Eberhart [28] proposed that an inertia coefficient η can be used in the velocity formula, mainly to maintain the motion inertia of the particles. Therefore, the first expression in the equation can be changed to ηv_i^k, and the setting of this coefficient can help expand the search space. Smaller values of η result in lesser inertial velocities, smaller search spaces, and the local optimization ability is relatively strong, although it is still likely to settle on a local optimal solution; when the value of η is larger, the inertial velocity is more significant, and the search space is more prominent. The global optimization ability

is relatively strong, but the convergence speed could be faster, and it is not easy to obtain the optimal solution. The PSO algorithm's ability to optimize can be adjusted by including the η variable.

Each particle updates its speed and position in each iteration according to Equations (11) and (12), and each particle moves in the search space according to the new position and speed. The particle will repeat the above behavior until it converges to the best solution.

When each particle updates its position information x_i^{k+1}, the value of its fitness function $F(x_i^{k+1})$ is calculated and compared with its fitness function value $F(x_{besti}^k)$ at the historical best position. If $F(x_i^{k+1}) \geq F(x_{besti}^k)$, that is, the fitness function value of the current particle is greater than or equal to its fitness function value at the historical best position, then update the historical best position of the particle and assign it as (x_i^{k+1}); otherwise, the historical best position of the particle remains unchanged. The following equation can be used to determine the historical best position calculation of particles:

$$p_{besti}^{k+1} = \begin{cases} p_{besti}^k; F(x_i^{k+1}) < F(p_{besti}^k) \\ x_i^{k+1}; F(x_i^{k+1}) \geq F(p_{besti}^k) \end{cases} \tag{13}$$

The global best position of the particle is updated according to the comparison between the adaptive function value $F(p_{besti}^{k+1})$ of the best position in the history of the particle and the adaptive function value $F(g_{best})$ of the global best position. If $F(p_{besti}^{k+1}) \leq F(g_{best})$, the global best position g_{best} is updated as follows:

$$g_{best} = \begin{cases} p_{besti}^k; F(p_{besti}^{k+1}) \leq F(g_{best}) \\ g_{best}; F(p_{besti}^{k+1}) > F(g_{best}) \end{cases} \tag{14}$$

2.3.2. Task Scheduling Algorithm Based on the PSO Algorithm

The PSO algorithm is applied to the task scheduling of a cloud computing system and executed as Algorithm 1.

Algorithm 1 PSO Algorithm

Input: m d-dimensional particle swarm S
Output: Particle swarm S with updated velocity and position information

repeat
  for each particle $i = 1, 2, ..., m$ **do** {*For each particle, calculate its fitness value and compare it with the fitness value of the historical best position.*}
    if $F(x_i^{k+1}) \leq F(p_{besti}^k)$ **then**
      $p_{besti}^{k+1} = x_i^{k+1}$
    else
      $p_{besti}^{k+1} = p_{besti}^k$
    end if
    if $F(p_{besti}^k) \leq F(g_{best})$ **then** {*Compare the fitness value of the current particle's historical best position with the fitness value of all particles' historical best positions.*}
      $g_{best} = p_{besti}^{k+1}$
    end if
    Update the velocity information of the particle using Equation (11)
    Update the position information of particles using Equation (12)
  end for
until Satisfy the termination condition (the number of iterations reaches the maximum, or the accuracy meets the requirements)

2.3.3. Algorithm Analysis

The PSO algorithm has parallel processing capability and relatively high efficiency. The main reason is that in the particle swarm, each particle has no dependency and can be

processed in parallel. In addition, in the PSO algorithm, the update of the particle's best position depends on the best historical position of the particle and the global historical best position. Despite the PSO algorithm's relatively quick convergence speed, it has the following drawbacks when it comes to workflow task scheduling problems:

1. When the PSO algorithm is scheduling tasks, it does not consider the calculation and data transmission of workflow tasks and can achieve better performance for independent task scheduling. However, the performance under workflow task scheduling is not satisfactory.
2. In the PSO algorithm, the optimization direction of the particle depends on its best historical position and the global best position, the convergence speed is fast, and it is easy to fall into the local optimal solution. Particles update their location based on their historical and neighborhood information, which can easily make the load on a single resource node too high, resulting in system instability.

3. PSO Algorithm Optimization

3.1. Particle Encoding Method

The particle encoding approach uses an indirect encoding scheme, and the particle's position data are used to determine how task resources should be distributed. The number of particles and their dimensions are equivalent to the number of tasks and the available resources, respectively. When there are n tasks to be scheduled in the cloud system having m resources, there are m particles with a dimension n; for the position information of each particle in the particle swarm, use $x_i = \{x_{i1}, x_{i2}, ..., x_{in}\}; \forall 1 \leq i \leq m$ means that each x_i represents a feasible solution of the PSO algorithm, while x_{i1} represents that the ith task is assigned to the resource x_{i1}. The speed of particles is expressed as $v_i = \{v_{i1}, v_{i2}, ..., v_{in}\}; \forall 1 \leq i \leq m$. Considering that there are seven particles and that the particle position information is $(7, 5, 4, 3, 6, 2, 1, 2)$, which means that the particle has eight dimensions, Table 1 shows the task resource allocation for the particle scheme.

Table 1. Scheme for allocating tasks.

Task	1	2	3	4	5	6	7	8
Resource	7	5	4	3	6	2	1	2

3.2. Improvements to the Initialization of Particles

The PSO algorithm randomly generates the position and velocity of the particles at the beginning, resulting in relatively low-quality particles. Based on the purpose of minimizing the task completion time in this chapter, this section uses the amount of data that needs to be calculated during task execution to construct a high-quality initial particle swarm. First, the computing performance of resources and the amount of data that must be processed during task execution are normalized to facilitate the subsequent selection of the most appropriate resource allocation resources.

Let Ur_{jcomp} be the normalized result of the computing capability of the jth resource. The closer the value is to 1, the higher the computing capability; the closer to 0, the lower the computing capability. $Ut_{ilength}$ represents the normalized result of the data that need to be calculated when the ith task is executed. The closer the value is to 1, the more significant the amount of data that needs to be calculated; the closer to 0, the smaller the amount that needs to be calculated. The values of Ur_{jcomp} and $Ut_{ilength}$ are between 0 and 1 and calculated as follows:

$$Ur_{jcomp} = \left| \frac{r_{jcomp} - \min(r_{comp})}{\max(r_{comp}) - \min(r_{comp})} \right| \tag{15}$$

$$Ut_{ilength} = \left| \frac{t_{ilength} - \min(t_{length})}{\max(t_{length}) - \min(t_{length})} \right| \tag{16}$$

where r_{jcomp} represents the computing performance of the jth computing resource; $t_{ilength}$ represents the amount of data that needs to be calculated when the ith task is executed. $\max(r_{comp})$ and $\min(r_{comp})$ respectively represent the computing power of the resource with the highest computing power and the resource with the lowest computing power in the combination of computing resources; $\max(t_{length})$ and $\min(t_{length})$ respectively represent the amount of data that needs to be executed in the task set of the most extensive task and the task with the smallest amount of data to be executed. The particle initialization process is shown in Figure 2.

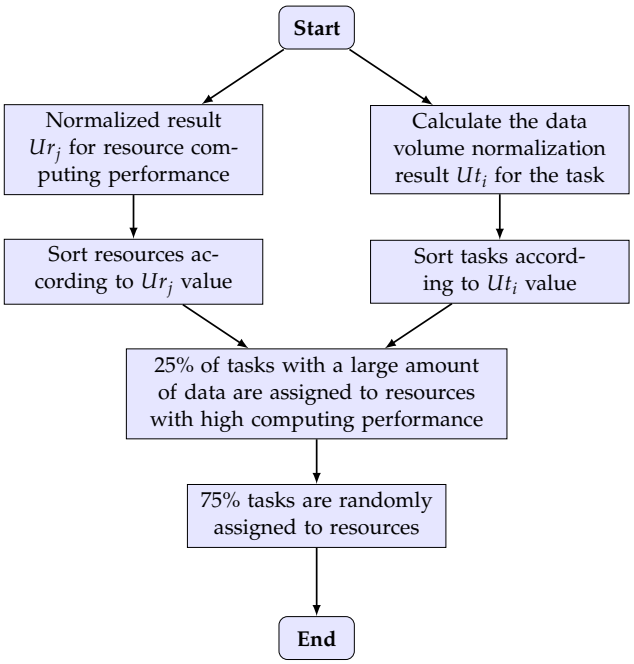

Figure 2. Particle initialization process.

According to Figure 2, the computing resources and tasks are sorted according to their Ur_{jcomp} and $Ut_{ilength}$ values. For 25% of tasks in the task set, the computing resources with higher computing performance are allocated to the tasks with higher computing performance requirements. For tasks with a relatively large amount of calculated data, a random allocation strategy is adopted for the remaining 75% of tasks. Adopting this allocation method can ensure the algorithm has a larger search space while ensuring the quality of the particles.

3.3. Adaptive Function Design

The adaptive function is a standard for assigning tasks to virtual machines. Its value is used to judge whether the tasks and resources are appropriate to perform subsequent operations according to the results. When defining the adaptive function, it is necessary to consider this paper's optimization goals: time cost and algorithm convergence.

According to Equation (10), set the adaptive function $F'(x)$ to ensure that the particle optimization process meets the user's requirements for the completion time of task execution. The adaptive function $F'(x)$ is calculated as follows:

$$F'(x) = \frac{1}{makespan} \tag{17}$$

The adaptive function $F'(x)$ is equal to the reciprocal of the task execution completion time so that the optimization direction of the particles is the same as the increasing direction of the adaptive function $F'(x)$.

3.4. Workflow Task Model Processing

The directed acyclic graph $G = (T, E)$ represents the evaluation of the task model in cloud computing and judges whether the model is computing-intensive or IO-intensive so that the model can be processed later.

For the set of available resources, its average computing power and average data transmission capacity are:

$$\overline{comp} = \frac{\sum_{j=1}^{m} r_{jcomp}}{m} \tag{18}$$

$$\overline{bw} = \frac{\sum_{j=1}^{m} r_{jbw}}{m} \tag{19}$$

For the task model $G = (T, E)$, the total calculation data volume $total\_length$ and the total transmission data volume $total\_data$ are calculated as follows:

$$total\_length = \sum_{i=1}^{n} t_{ilength} \tag{20}$$

$$total\_data = \sum_{i=1}^{n} \sum_{j=i+1}^{n} e_{ij} \tag{21}$$

According to Equations (18)–(21), the average execution time of all tasks and the average calculation time of tasks are calculated as follows:

$$comp\_time = \frac{total\_length}{\overline{comp}} \tag{22}$$

$$trans\_time = \frac{total\_data}{\overline{bw}} \tag{23}$$

Take the condition $comp\_time \geq trans\_time$; if this condition is met, the task is a computing-intensive task; otherwise, it is an IO-intensive task.

3.4.1. Computationally Intensive Tasks

The graph G for computationally intensive tasks is simplified according to the following steps.

1. Calculate the sum of the node weights on the path from the root node to each leaf node;
2. "Gather" the path with the largest calculated value above the root node, and update the weight information of the root node;
3. "Remove" the root node from the graph;
4. Repeat the above process until all the nodes in the graph are "independent nodes" without successor nodes.

According to the above process, the workflow task is converted into an independent task. Several independent tasks contain several subtasks. Due to the tremendous amount of calculation, they are gathered together. When allocating resources in the later stage, a whole task is allocated to the same resource node with better performance, which reduces the execution time to a certain extent.

3.4.2. IO-Intensive Tasks

The simplification of the graph G for IO-intensive tasks is similar to that of computationally intensive tasks, and the simplification is performed according to the following steps.

1. Calculate the sum of weights on the path from the root node to each leaf node;
2. "Gather" the path with the largest calculated value above the root node, and update the weight information of the root node;

3. "Remove" the root node from the graph;
4. Repeat the above process until all the nodes in the graph are "independent nodes" without successor nodes.

The processing of IO-intensive tasks is similar to that of computing-intensive tasks. The difference is that nodes with relatively large data transmissions are gathered together. When resources are allocated later, they are allocated to one resource as a whole to reduce the impact of data transmission time on execution completion time.

Workflow task model processing combines intensive tasks into one particle, which reduces the particle dimension and simplifies the optimization complexity of the subsequent PSO algorithm.

3.4.3. Algorithm Implementation

This paper improves the PSO algorithm's task scheduling, and the proposed algorithm is called Enhanced Particle Swarm Optimization (EPSO). The algorithm execution steps are as Algorithm 2.

Algorithm 2 EPSO Algorithm

1: According to the directed acyclic graph describing the task and the condition $comp\_time \geq trans\_time$, it is judged whether the task is computationally intensive or IO-intensive;
2: According to the different types of tasks, the tasks are processed by the workflow task model to obtain several sets of "independent tasks" (Section 3.4);
3: Use the set of "independent tasks" as input to the algorithm;
4: Initialize the particles;
5: **repeat**
6: Traverse the particle swarm;
7: Calculate the adaptive function value of the particle according to Equation (17);
8: Update the historical best position p_{best} and the global best position g_{best} of the particle according to Equations (13) and (14);
9: Update the particles' position and velocity information according to Equations (11) and (12);
10: **until** the number of iterations reaches the maximum

4. Experiments and Analysis

The proposed EPSO algorithm has the advantage of workflow task scheduling, which can effectively reduce the scheduling time of workflow tasks and obtain lower scheduling costs. This section compares the EPSO algorithm with the HPSO algorithm proposed in Pandey et al. [8] to verify the superiority of the EPSO algorithm in terms of time and cost (computation cost and data transmission cost). These algorithms are tested under applications containing 10, 20, 50, 100, 150, 200, 250, and 300 workflows and independent tasks. Each algorithm was run 20 times under different task types and different task numbers, and the average value was taken as the experimental result. This experiment uses the cloud environment simulation tool CloudSim to simulate the cloud environment. The physical node configuration uniformly adopts the configuration of an HP ProLiant DL380 G9 server (Xeon E5-2667v3, 8 cores), 64 GB. The relevant parameters are initialized as shown in Table 2.

Table 2. Algorithm/CloudSim parameter settings.

Parameter	Value
Maximum number of resources	250
Particle inertia	0.99
The maximum number of iterations	50
Task length	10,000–40,000
The amount of data transferred by the task	150–200
Task computing requirements	500–1000
Task bandwidth requirements	60–100

For costs computation, we used Amazon EC2's pricing guidelines for various classes of virtual machine instances while varying the processing costs and assuming one hour for the completion of all tasks. The communication costs are comparable to what Amazon CloudFront charges for each unit of data sent between resources. The execution costs are computed according to Vecchiola et al. [29] while shifting the size of the tasks.

4.1. Workflow Task Scheduling

This section presents the experimental results of the compared algorithms for scheduling workflow tasks with different numbers of tasks in terms of computation time and cost.

4.1.1. Time

Table 3 shows the experimental data of the time spent by the EPSO algorithm, the HPSO algorithm, and the PSO algorithm for scheduling workflow tasks with different numbers of tasks.

Table 3. Time performance under different number of workflow tasks (unit: seconds).

Number of Tasks	EPSO	HPSO	PSO
10	102.485	104.485	105.416
20	110.215	115.475	116.125
50	124.596	139.156	145.156
100	170.123	198.417	221.152
150	196.482	232.156	265.478
200	204.151	256.482	298.985
250	235.156	354.545	398.121
300	255.145	394.156	459.562

Figure 3 shows the line chart of Table 3. When the number of tasks is less than 50, there is no significant difference in the performance of the three algorithms. This is mainly because when the number of tasks is relatively small, the search performance of the three algorithms is equivalent, and the best solution can be obtained in a limited search space. The EPSO algorithm shows significantly superior performance as the number of tasks increases. This is mainly because the EPSO algorithm performs workflow task model processing on workflow tasks, effectively reducing the dimension of particles in the particle swarm and making the algorithm optimization process faster. In addition, the processing of the workflow model enables intensive tasks to be aggregated and assigned to appropriate resource nodes, reducing the time needed to complete task execution. At the same time, the initialization operation of the EPSO algorithm on the particle swarm makes it have high-quality particles, ensuring that the particles have a relatively high quality. The EPSO algorithm can find a better solution than the other two algorithms.

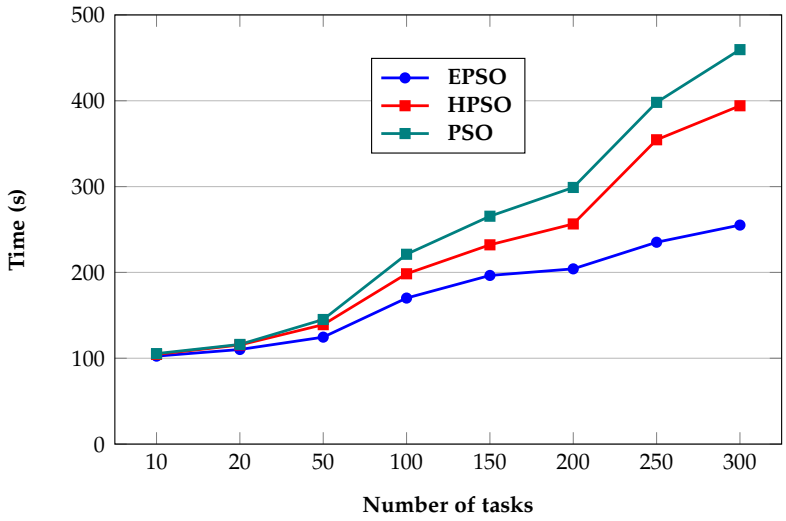

Figure 3. A graphical representation of the results given in Table 3.

4.1.2. Cost

Figure 4 shows the cost of the EPSO algorithm, HPSO algorithm, and PSO algorithm for workflow task scheduling under different task numbers, and Table 4 shows the experimental data in Figure 4.

Table 4. Cost performance under different number of workflow tasks (unit: US dollars).

Number of Tasks	EPSO	HPSO	PSO
10	11.5	11.3	12.1
20	24.3	22.5	25.9
50	56.3	59.4	77.4
100	167.6	175.1	180.2
150	218.1	246.2	286.9
200	307.2	335	350.8
250	359.1	397.4	439.5
300	401.5	459.8	558.3

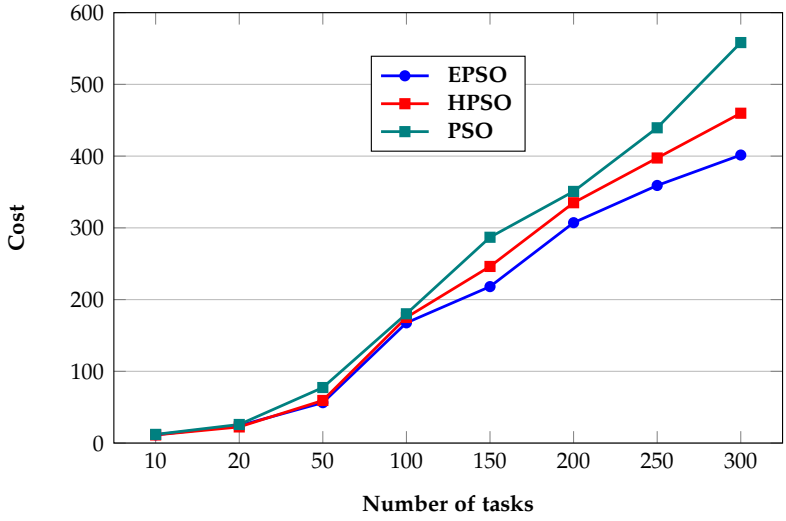

Figure 4. A graphical representation of the results given in Table 4.

There is not much of a cost difference between the algorithms when the number of tasks is limited, but it can still be seen that the cost of the EPSO algorithm is slightly lower than the other two. With the gradual increase in the number of tasks, the EPSO algorithm is

superior to other algorithms. The main reason is that the EPSO algorithm has advantages in workflow task scheduling.

4.2. Independent Task Scheduling

4.2.1. Time

Table 5 shows the experimental data of the time spent by the EPSO algorithm, HPSO algorithm, and PSO algorithm on the scheduling of independent tasks with different numbers of tasks, and Figure 5 shows the line chart of Table 5.

According to Figure 5, it can be seen that when dealing with independent tasks, the EPSO algorithm is slightly better than the HPSO algorithm and obviously better than the PSO algorithm; that is, the EPSO algorithm can also achieve good performance when dealing with independent tasks. This is mainly due to the initial processing of the particles by the EPSO algorithm to obtain high-quality initial particles. The design of the adaptive function also ensures the optimal direction of the particles and improves the quality of the feasible solution.

Table 5. Time performance under different number of independent tasks (unit: seconds).

Number of Tasks	EPSO	HPSO	PSO
10	72.325	83.534	92.785
20	81.563	95.265	108.783
50	99.125	114.255	127.678
100	127.425	154.678	189.425
150	151.897	178.787	217.454
200	178.578	212.673	259.425
250	198.435	226.524	287.542
300	225.523	254.246	327.789

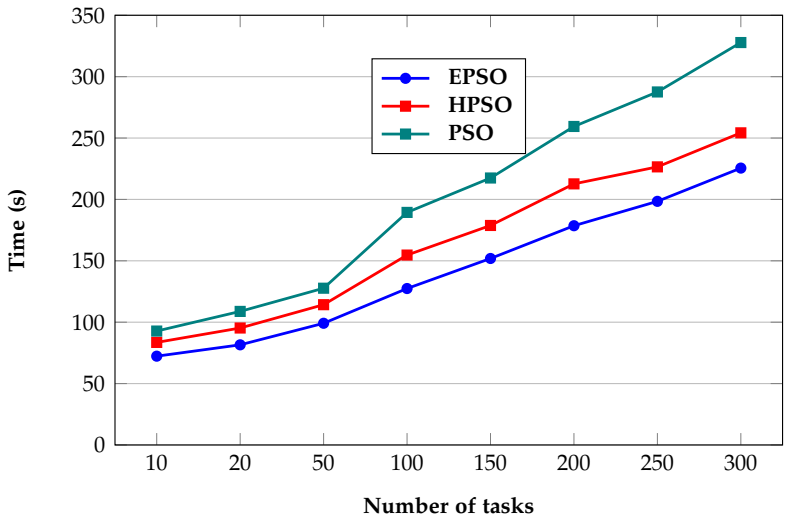

Figure 5. A graphical representation of the results given in Table 5.

4.2.2. Cost

Table 6 shows the experimental data of EPSO algorithm, HPSO algorithm, and PSO algorithm on the cost of different task numbers, and Figure 6 shows the results of Table 6.

It can be seen from Figure 6, similar to the processing of workflow tasks, that the cost of the EPSO algorithm is lower than the other two algorithms. While the EPSO algorithm achieves a shorter completion time, it also optimizes the cost to a certain extent.

Table 6. Cost performance under different numbers of independent tasks (unit: US dollars).

Number of Tasks	EPSO	HPSO	PSO
10	8.2	9.3	11.9
20	21.7	22.4	26.8
50	49.1	57.4	72.3
100	142.8	167.7	196.4
150	214.7	239.1	290.9
200	298.3	325.7	389.5
250	353.3	382.4	457.5
300	391.5	432.8	554.7

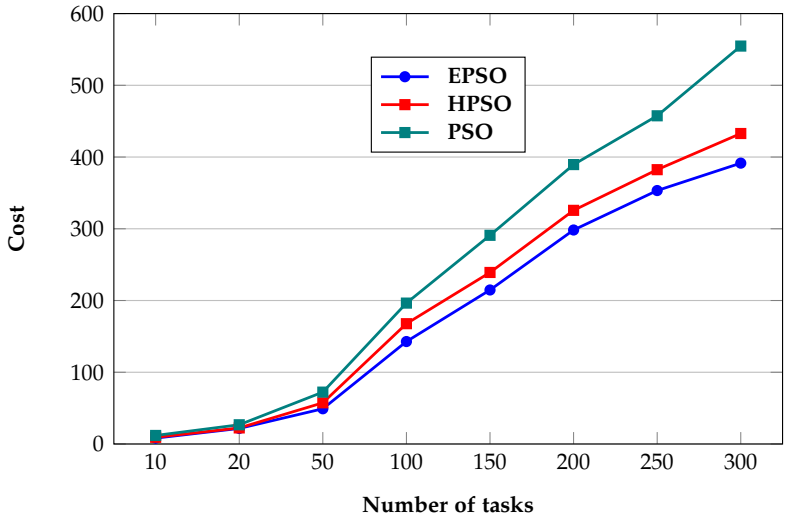

Figure 6. A graphical representation of the results given in Table 6.

4.3. Convergence

In order to verify the convergence of the EPSO algorithm, a workflow containing 100 tasks was selected to increase the number of iterations. The EPSO algorithm was run, and the completion time was recorded. The results are shown in Figure 7.

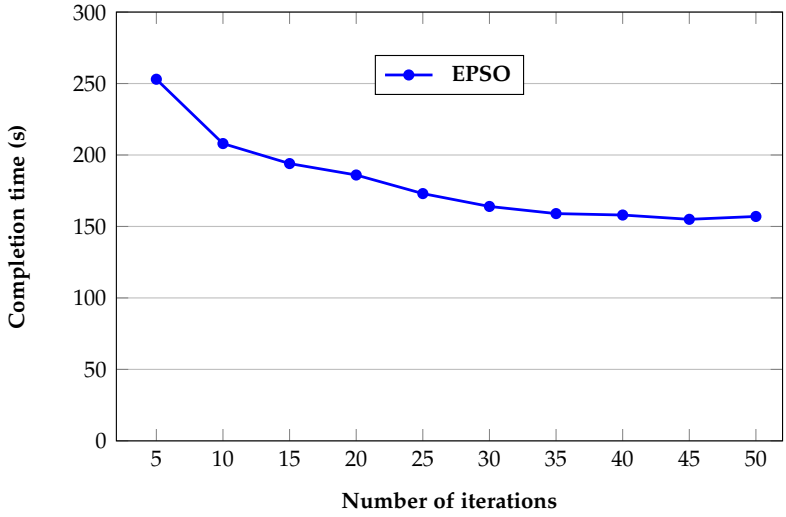

Figure 7. Convergence of the EPSO algorithm when the number of iterations increases.

As seen in Figure 7, the EPSO algorithm has better convergence, primarily because of the reduction in particle dimensions when the number of iterations is more than 25, and the completion time does not alter with the increase in iterations. When the algorithm is

first executed, a high-quality particle swarm is ensured by optimizing the initial position information of the particles, and an effective optimization direction is ensured by using the proper adaptive function design. This improves the quality, speed, and convergence of the EPSO algorithm.

It can be seen from the above experiments that the EPSO algorithm shows good convergence in the solution process, can obtain high-quality solutions in a short time, and has an indelible advantage in workflow task scheduling.

5. Conclusions

In this paper, based on previous research on the task scheduling of the PSO algorithm and taking into account the problem of time cost in the case of workflow task processing, a workflow task model processing algorithm was proposed, putting forward the optimization of the PSO algorithm under the condition of optimizing the execution time and execution cost. Through workflow task model processing, particle dimensionality is reduced, and tasks with large amounts of data are merged. The algorithm's search space and convergence speed are guaranteed by improving the particle initialization operation. The problem of time constraints in task scheduling is solved by improving the adaptive function. Experiments clearly showed that the proposed EPSO algorithm is effective in reducing the time and the cost.

The following areas of future work will receive the majority of attention:

- The adaptive function can be improved according to different application scenarios in the future.
- The optimization target can be broadened in order to consider other important objectives, such as load balancing, CPU utilization, and energy consumption.
- The integration and evaluation of the proposed algorithm in the context of a real-world unified cloud management platform can be investigated.

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: S.H.A. and M.A.R.; data collection: S.H.A. and M.A.R.; analysis and interpretation of results: S.H.A. and M.A.R.; draft manuscript preparation: S.H.A. and M.A.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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 conflict of interest.

References

1. Marston, S.; Li, Z.; Bandyopadhyay, S.; Zhang, J.; Ghalsasi, A. Cloud computing—The business perspective. *Decis. Support Syst.* **2011**, *51*, 176–189. [CrossRef]
2. Hoffa, C.; Mehta, G.; Freeman, T.; Deelman, E.; Keahey, K.; Berriman, B.; Good, J. On the use of cloud computing for scientific workflows. In Proceedings of the 4th IEEE International Conference on eScience, eScience 2008, Indianapolis, IN, USA, 7–12 December 2008; pp. 640–645. [CrossRef]
3. Chenhong, Z.; Shanshan, Z.; Qingfeng, L.; Jian, X.; Jicheng, H. Independent tasks scheduling based on genetic algorithm in cloud computing. In Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, WiCOM 2009, Beijing, China, 24–26 September 2009. [CrossRef]
4. Kennedy, J. Particle Swarm Optimization. In *Encyclopedia of Machine Learning*; Sammut, C.; Webb, G.I., Eds.; Springer: Boston, MA, USA, 2011; pp. 760–766. [CrossRef]
5. Xue, S.J.; Wu, W. Scheduling Workflow in Cloud Computing Based on Hybrid Particle Swarm Algorithm. *Telkomnika Indones. J. Electr. Eng.* **2012**, *10* . [CrossRef]
6. Guo, L.; Zhao, S.; Shen, S.; Jiang, C. Task scheduling optimization in cloud computing based on heuristic Algorithm. *J. Netw.* **2012**, *7*, 547–553. [CrossRef]
7. Varalakshmi, P.; Ramaswamy, A.; Balasubramanian, A.; Vijaykumar, P. An Optimal Workflow Based Scheduling and Resource Allocation in Cloud. In *Advances in Computing and Communications*; Abraham, A., Lloret Mauri, J., Buford, J.F., Suzuki, J., Thampi, S.M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 411–420.

8. Pandey, S.; Wu, L.; Guru, S.M.; Buyya, R. A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In Proceedings of the International Conference on Advanced Information Networking and Applications, AINA, Perth, WA, Australia, 20–23 April 2010; pp. 400–407. [CrossRef]
9. Arunarani, A.R.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
10. Motlagh, A.A.; Movaghar, A.; Rahmani, A.M. Task scheduling mechanisms in cloud computing: A systematic review. *Int. J. Commun. Syst.* **2020**, *33*, e4302. [CrossRef]
11. Bulchandani, N.; Chourasia, U.; Agrawal, S.; Dixit, P.; Pandey, A. A survey on task scheduling algorithms in cloud computing. *Int. J. Sci. Technol. Res.* **2020**, *9*, 460–464.
12. Ibrahim, I.M.; Zeebaree, S.R.M.; M.Sadeeq, M.A.; Radie, A.H.; Shukur, H.M.; Yasin, H.M.; Jacksi, K.; Rashid, Z.N. Task Scheduling Algorithms in Cloud Computing: A Review. *Turk. J. Comput. Math. Educ. (TURCOMAT)* **2021**, *12*, 1041–1053. [CrossRef]
13. Houssein, E.H.; Gad, A.G.; Wazery, Y.M.; Suganthan, P.N. Task Scheduling in Cloud Computing based on Meta-heuristics: Review, Taxonomy, Open Challenges, and Future Trends. *Swarm Evol. Comput.* **2021**, *62*, 100841. [CrossRef]
14. Awad, A.I.; El-Hefnawy, N.A.; Abdel-Kader, H.M. Enhanced Particle Swarm Optimization for Task Scheduling in Cloud Computing Environments. *Procedia Comput. Sci.* **2015**, *65*, 920–929. [CrossRef]
15. Mirzayi, S.; Rafe, V. A hybrid heuristic workflow scheduling algorithm for cloud computing environments. *J. Exp. Theor. Artif. Intell.* **2015**, *27*, 721–735. [CrossRef]
16. Xue, S.; Shi, W.; Xu, X. A Heuristic Scheduling Algorithm based on PSO in the Cloud Computing Environment. *Int. J. u- e- Serv. Sci. Technol.* **2016**, *9*, 349–362. [CrossRef]
17. Huang, X.; Li, C.; Chen, H.; An, D. Task scheduling in cloud computing using particle swarm optimization with time varying inertia weight strategies. *Clust. Comput.* **2020**, *23*, 1137–1147. [CrossRef]
18. Alsaidy, S.A.; Abbood, A.D.; Sahib, M.A. Heuristic initialization of PSO task scheduling algorithm in cloud computing. *J. King Saud Univ.—Comput. Inf. Sci.* **2022**, *34*, 2370–2382. [CrossRef]
19. Su, Y.; Bai, Z.; Xie, D. The optimizing resource allocation and task scheduling based on cloud computing and Ant Colony Optimization Algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2021**. [CrossRef]
20. Hamed, A.Y.; Alkinani, M.H. Task scheduling optimization in cloud computing based on genetic algorithms. *Comput. Mater. Contin.* **2021**, *69*, 3289–3301. [CrossRef]
21. Chaudhary, N.; Kalra, M.; Scholar, P.G. An improved Harmony Search algorithm with group technology model for scheduling workflows in cloud environment. In Proceedings of the 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics, UPCON 2017, Mathura, India, 26–28 October 2017; pp. 73–77. [CrossRef]
22. Gabi, D.; Ismail, A.S.; Zainal, A.; Zakaria, Z.; Al-Khasawneh, A. Hybrid cat swarm optimization and simulated annealing for dynamic task scheduling on cloud computing environment. *J. Inf. Commun. Technol.* **2018**, *17*, 435–467. [CrossRef]
23. Chen, X.; Long, D. Task scheduling of cloud computing using integrated particle swarm algorithm and ant colony algorithm. *Clust. Comput.* **2019**, *22*, 2761–2769. [CrossRef]
24. Jia, L.W.; Li, K.; Shi, X. Cloud Computing Task Scheduling Model Based on Improved Whale Optimization Algorithm. *Wirel. Commun. Mob. Comput.* **2021**, *2021*, 4888154. [CrossRef]
25. Keivani, A.; Tapamo, J.R. Task scheduling in cloud computing: A review. In Proceedings of the 2nd International Conference on Advances in Big Data, Computing and Data Communication Systems, Winterton, South Africa, 5–6 August 2019; pp. 1–6. [CrossRef]
26. Sharma, P.; Shilakari, S.; Chourasia, U.; Dixit, P.; Pandey, A. A survey on various types of task scheduling algorithm in cloud computing environment. *Int. J. Sci. Technol. Res.* **2020**, *9*, 1513–1521.
27. Shami, T.M.; El-Saleh, A.A.; Alswaitti, M.; Al-Tashi, Q.; Summakieh, M.A.; Mirjalili, S. Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access* **2022**, *10*, 10031–10061. [CrossRef]
28. Shi, Y.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the IEEE Conference on Evolutionary Computation, ICEC, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73. [CrossRef]
29. Vecchiola, C.; Kirley, M.; Buyya, R. Multi-Objective Problem Solving With Offspring on Enterprise Clouds. In Proceedings of the 10th International Conference on High Performance Computing in Asia-Pacific Region, Kaohsiung, Taiwan, 2–5 March 2009; pp. 132–139. [CrossRef]