applied sciences

MDPI

Article

# Dynamic Task Scheduling in Remote Sensing Data Acquisition from Open-Access Data Using CloudSim

**Zhibao Wang [1,2], Lu Bai [3], Xiaogang Liu [1], Yuanlin Chen [1], Man Zhao [4] and Jinhua Tao [5,6,*]**

[1] School of Computer and Information Technology, Northeast Petroleum University, Daqing 163318, China
[2] Bohai-Rim Energy Research Institute, Northeast Petroleum University, Qinhuangdao 066004, China
[3] School of Computing, Ulster University, Belfast BT15 1ED, UK
[4] School of Communication and Electronic Engineering, Qiqihaer University, Qiqihaer 161003, China
[5] State Key Laboratory of Remote Sensing Science, Aerospace Information Research Institute of Chinese Academy of Sciences, Beijing 100101, China
[6] University of Chinese Academy of Sciences, Beijing 100049, China
* Correspondence: taojh@radi.ac.cn

**Abstract:** With the rapid development of cloud computing and network technologies, large-scale remote sensing data collection tasks are receiving more interest from individuals and small and medium-sized enterprises. Large-scale remote sensing data collection has its challenges, including less available node resources, short collection time, and lower collection efficiency. Moreover, public remote data sources have restrictions on user settings, such as access to IP, frequency, and bandwidth. In order to satisfy users' demand for accessing public remote sensing data collection nodes and effectively increase the data collection speed, this paper proposes a TSCD-TSA dynamic task scheduling algorithm that combines the BP neural network prediction algorithm with PSO-based task scheduling algorithms. Comparative experiments were carried out using the proposed task scheduling algorithms on an acquisition task using data from Sentinel2. The experimental results show that the MAX-MAX-PSO dynamic task scheduling algorithm has a smaller fitness value and a faster convergence speed.

**Keywords:** remote sensing data; big data acquisition; task scheduling; PSO; CloudSim

check for updates

**Citation:** Wang, Z.; Bai, L.; Liu, X.; Chen, Y.; Zhao, M.; Tao, J. Dynamic Task Scheduling in Remote Sensing Data Acquisition from Open-Access Data Using CloudSim. *Appl. Sci.* **2022**, *12*, 11508. https://doi.org/10.3390/app122211508

Academic Editors: Anselme Muzirafuti, Dimitrios S. Paraforos, Giovanni Randazzo and Stefania Lanza

Received: 20 September 2022
Accepted: 8 November 2022
Published: 12 November 2022



## 1. Introduction

In recent years, with the rapid advancement of satellite remote sensing technology, the number of satellites launched by various countries across the world has been increasing year by year [1]. Low- and medium-resolution satellites used for ocean, atmosphere, environment, social welfare, and scientific research form a large proportion among them at the regional to global scale [2]. Research on massive remote sensing images utilizing deep learning models plays a significant part in urban planning [3,4], environmental observations [5,6], and energy development [7], especially with the recent increasing interest. The collection of data is time-consuming, labor-intensive, and difficult. The amount of data is undergoing explosive growth and increasing at a petabyte level. Taking the Landsat 8 OLI/TIRS Level-2 data type in the United States Geological Survey (USGS) [8] as an example, the China region alone generated about 1500 remote sensing data products per month, each with a data capacity of 2G, in 2019. Remote sensing technology has been unprecedentedly improved in terms of data acquisition capabilities and information service capabilities. In order to realize the sharing service of satellite remote sensing data, the National Aeronautics and Space Administration (NASA) [9] and European Space Agency (ESA) [10] have established remote sensing data-sharing portals for the above public satellites and provide each registered user with equal access to remote sensing data. Therefore, these data-sharing portals have gradually become a popular data source of large amounts of acquired remote sensing data for individuals, small, medium, and

micro enterprises, and scientific research institutions. Although these data-sharing portals provide each registered user with an equal opportunity to obtain remote sensing data, the limitations on the access to IP, frequency, and bandwidth are becoming challenges in the large-scale acquisition of remote sensing data for users. Google Earth Engine (GEE) is a very useful resource for large-scale remote sensing data applications [11,12]. Although it is free for research projects, it is limited in capacity for data export. It is also not suitable for uploading private datasets to GEE for processing [13]. GEE cannot satisfy large-scale remote sensing scientific research. Many remote sensing research activities, including remote sensing observations and data assimilation for Earth system models, are not supported by GEE. The rapid development of cloud technologies [14] can be used for processing and integrating large-scale, heterogeneous resources, which provides further opportunities for large-scale massive remote sensing data acquisition.

Early work on task scheduling with parallel computing was conducted by Sarkar in 1989 [15]. Task scheduling has always been a hot research topic in project production and scientific research, and it is also a major research area of interest to researchers. This type of problem is ultimately a non-deterministic polynomial problem (NP) for task scheduling. Generally, task scheduling algorithms consist of traditional task scheduling and meta-heuristic algorithms [16]. Traditional task scheduling algorithms include First Come First Serve (FCFS) [17], Shortest Job First (SJF) [17], Max-Min [18], Min-Min [19], and polling scheduling [20,21]. Maheswaran et al. [22] modified standard heuristics for task assignment in predictable environments. Meta-heuristic task scheduling algorithms include Particle Swarm Optimization (PSO) [23,24], a genetic algorithm (GA) [25], an ant algorithm [26], BP algorithms [27,28], and an adaptive meta-heuristic-based method [29]. Recently, some studies have focused on improved hybrid algorithms based on the above-mentioned algorithms. Elmougy et al. [30] proposed a novel hybrid algorithm that combines SJF and Round Robin (RR) schedulers considering a dynamic variable task quantum. Manasrah and Ali [31] proposed a hybrid GA-PSO algorithm to reduce the makespan and balance the load of dependent tasks over heterogeneous resources. Choudhary et al. [32] proposed an efficient hybrid algorithm combining the gravitational search algorithm and heterogeneous earliest finish time algorithm.

With the significantly increasingly large amounts of data produced by satellites, more studies have been conducted to address the challenges in remote sensing big data storage [33,34] and the efficient and scalable processing of remotely sensed data [1,35–37]. PSO is widely used for different remote task scheduling research. An et al. [38] used PSO in task scheduling for unmanned aerial vehicle (UAV) swarm remote sensing in distributed photovoltaic array maintenance. Wu et al. [39] proposed a hybrid algorithm of PSO and a genetic algorithm to improve the dynamic regional splitting planning of a remote sensing satellite swarm. There is a growing need to develop an effective multi-objective scheduling framework that jointly minimizes the task cost, task energy consumption, and execution time. Alkayal et al. [40] developed a multi-objective PSO algorithm to minimize the waiting time and maximize the system throughput. Gabi et al. [41] proposed a multi-objective Quality-of-Service model to address customers' expectations based on execution time and execution cost criteria. Xing et al. [42] proposed a comprehensive multi-objective model to solve a task scheduling sub-problem by using the Bayes belief model and learnable ant colony optimization. Chen et al. [43] used an evolutionary computation scheduling method for satellite periodic continuous observation task scheduling, and several constraints were considered for the satellite working pattern. In a recent study by Sun et al. [44], an energy-efficient solution for multi-objective task scheduling for the cloud implementation of hyperspectral image classification was proposed. However, only execution time and energy consumption were taken into consideration.

With the emergence of new types of Internet of Things (IoT) devices, the performance service requirements for the network have increased. In addition to the traditional way of using upgraded network bandwidth to avoid network congestion, it can also predict the available performance level of the network by analyzing historical service data so as

to select network resources and make adjustments according to actual conditions. Based on the historical data of the network transmission of nodes, it is found that there are relatively few methods to study the law of the network transmission of nodes. Especially in the field of remote sensing data collection, there is little relevant research on predicting the collection speed of each node, guiding dynamic task scheduling, and improving the resource utilization of nodes. This paper mainly studies network-intensive task scheduling. The research on network data transmission problems is mostly aimed at the transmission of a large amount of task data or the higher requirements for network transmission data. In order to reduce the impact of Coflow Completion Time (CCT) on applications in the data parallel framework, in 2018, Yangming Zhao [45] changed the practice of pre-setting coflow composing flow endpoints in traditional research work and proposed a joint online Reducer Placement and Coflow bandwidth scheduling framework to minimize the average CCT in the cloud cluster. In 2019, Duggan et al. [46] used the recurrent neural network sequence prediction algorithm to realize the prediction of the central processing unit (CPU) utilization and network bandwidth utilization of real-time migration in the cloud environment. In order to monitor the network traffic of the network data center, network utilization must be improved. The conventional approach is to modify the hardware or terminal host to increase monitoring resources, but this will increase monitoring overhead and is not conducive to long-term development. Chao et al. [47] proposed a fast, low-overhead image flow monitoring and scheduling system called FlowSeer for data flow mining. The model can accurately and quickly predict the speed and duration of any initial data flow and meet the needs of the dynamic adjustment of data flow-routing strategies.

Many research methods on network performance prediction and network-intensive task scheduling are discussed above, but most of these methods aim to improve the internal execution efficiency of the network, avoid data collisions by scheduling time slots, and achieve the purpose of reducing the network data delay. In order to improve the resource utilization of the collection nodes of remote sensing data collection users and effectively improve the collection speed of a large amount of remote sensing data, this paper uses the backpropagation (BP) neural network algorithm to predict the collection speed of the remote sensing data collection node. The dataset used in this work was obtained from Sentinel2, and the selected area is the whole China Region in 2019 [10]. According to the difference in the speed of the collection task performed by each node at different times, a two-stage combined dynamic task scheduling algorithm (TSCD-TSA) is proposed to match the task set and the resource node set, making dynamic adjustments to achieve the aim of improving the utilization of resource node sets. Comparative experiments were designed to evaluate the performance of applying different improved dynamic task scheduling algorithms, including FCFS-PSO, SJF-PSO, MAX-MAX-PSO, and PSO. In order to evaluate the effectiveness of the proposed optimized task scheduling algorithms, the CloudSim platform was used to evaluate the algorithms. The execution results of different task sets were analyzed.

The rest of this paper is organized as follows: In Section 2, methods for task scheduling and our proposed optimized task scheduling algorithms are introduced. In Section 3, the experiment design to compare the performance of the different proposed optimized task scheduling algorithms is described, and the experimental results are presented and discussed. Finally, the conclusion is drawn in Section 4.

## 2. Methods

Given the massive amount of public remote sensing data collected using limited resource nodes, this paper is mainly focused on studying the large-scale public remote sensing matching problem between data collection tasks and collection nodes. First, a model of multi-objective task scheduling is proposed with the consideration of time limit constraints, energy consumption constraints, cost constraints, and the task scheduling objective function. Then, an improved multi-objective PSO optimization algorithm is proposed to further improve the allocation efficiency. Finally, multiple groups of simulations

for comparative experiments were designed to compare the task execution time, energy consumption, and cost of various allocation schemes under various task scheduling.

The task scheduling architecture for conducting public remote sensing data collection is shown in Figure 1. A web crawler is used to obtain remote sensing data task Uniform Resource Locators (URLs) in batches from public remote sensing data sources and add them to the task pool. The scheduler selects a task set of an appropriate size from the task pool to match the execution resource set and outputs the corresponding scheduling scheme. Each executor selects related tasks for execution in strict accordance with the respective task sequence in the scheduler's output task scheduling plan, and the prediction module predicts the future execution speed based on the historical execution records of each executor. The scheduler dynamically adjusts the current task scheduling scheme according to the prediction results of the prediction module and the task execution of each actuator (it only reassigns the tasks that have not been executed in the task set; the tasks that have been executed in the task set are not within the scope of rescheduling). For tasks completed by each actuator, for example, the quality of remote sensing data collected by each actuator needs to be verified. Qualified data will then be entered into the storage center, while unqualified tasks need to be re-collected.

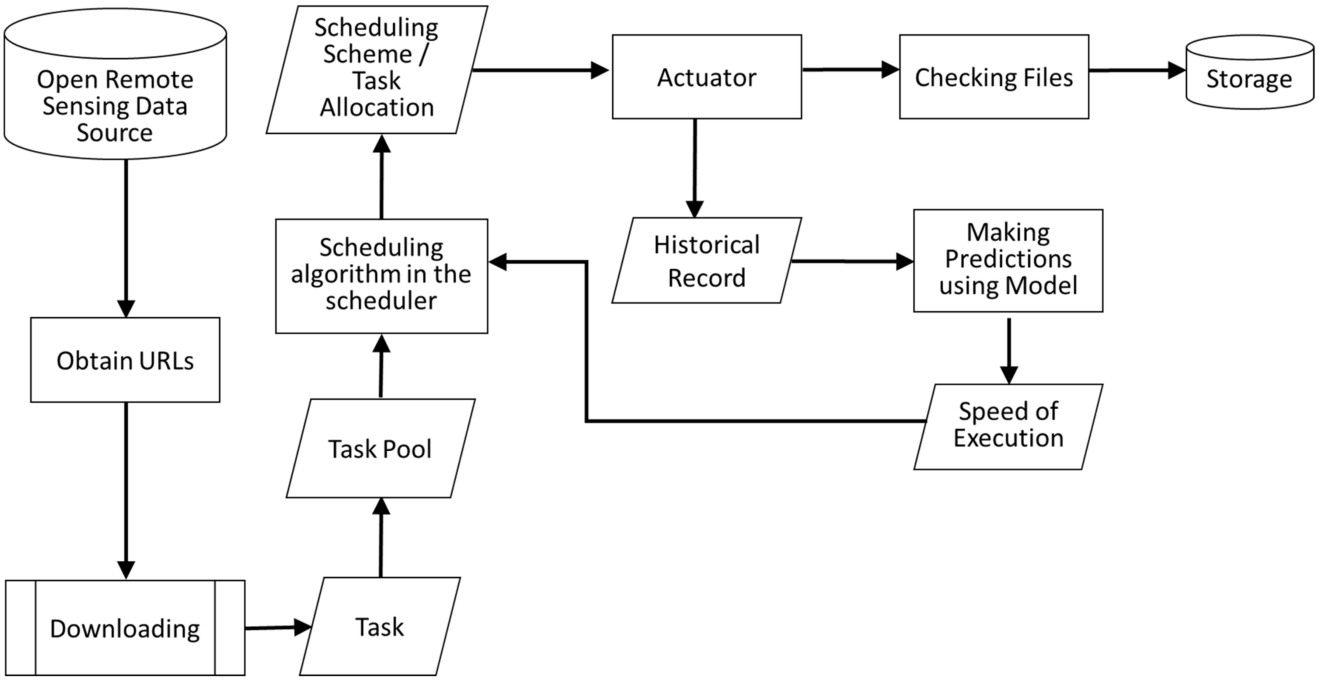


**Figure 1.** Task scheduling architecture diagram.

According to the scheduling architecture of the open remote sensing data acquisition task in Figure 1, the following assumptions are made during the execution of the task:

- The actuators perform tasks independently and do not interfere with each other.
- The choice of the task set size in the task pool is defined by the user, and the number of task sets is usually much larger than the tasks being executed. Users can select an appropriate number of task sets according to the size of each task in the task set and the execution speed of each actuator so as to avoid some time-limited tasks being in a waiting state for a long time.
- If the task URL has not been validated, each task URL will only be executed by an executor and only once; that is, there is no execution conflict in the task.
- In order to speed up the training, the prediction module adopts the strategy of incremental learning of the execution history.

- The update frequency of the scheduling algorithm in the scheduler is similar to that of the prediction module. After the prediction module produces new prediction results, task rescheduling is started immediately.

### 2.1. Task Scheduling Model

#### 2.1.1. Problem Statement

It is assumed that the task set is $T = \{t_1, t_2, \ldots, t_m\}$ and the node set is $W = \{w_1, w_2, \ldots, w_n\}$. Each task in set $T$ is expressed as $t_i (1 \leq i \leq m)$, and the task scheduling algorithm will match task $t_i$ with node $w_j (1 \leq j \leq n)$ according to the collection ability of each collection node in collection node set $W$. The result of static task scheduling can be expressed by $\{< t_1,\ w_1 >, < t_2, w_2 >, \ldots, < t_i, w_j >\}$. Dynamic task scheduling aims to dynamically adjust the matching sequence of tasks and nodes, of which the scheduler dynamically reschedules tasks that have not yet been executed according to the current task execution speed of each node.

#### 2.1.2. Multi-Objective Optimization Model

Multi-objective optimization, also known as multi-criteria optimization, whose aim is to optimize multiple sub-objectives at the same time, is considered to be a hot research topic in the field of mathematics and multi-criteria analysis. When a sub-objective is being optimized by multiple objectives, it may affect the target performance of other sub-objectives. Therefore, there is a nonlinear relationship among multiple optimization sub-objectives, which makes it difficult to optimize multiple sub-objectives at the same time.

According to the Pareto optimal solution principle, when there are multiple optimization objectives in a problem at the same time, there are conflicts and incomparable phenomena between standards for the multiple sub-objectives. The solution of the problem may be optimal for one goal, but not for other goals. When solving the objective function of this type of problem, it will inevitably weaken the solution for other objectives. Therefore, this principle is called the non-dominated solution or Pareto solution. The solution to this kind of problem is to convert the corresponding mathematical model to a problem with multi-objective function solutions. The maximal problem with an m-dimensional decision vector and n optimization objectives is usually defined as:

$$Min\{F(x) = [f_1(x),\ f_2(x),\ \ldots,\ f_n(x)]\} \tag{1}$$

in which there is a decision space $Q$ and a target space $R$, and there are m-dimensional decision variables $x = (x_1,\ x_2, \ldots, x_m) \in Q$ and an n-dimensional target vector $(f_1,\ f_2, \ldots, f_n) \in R$. The solution process is to map the decision space through the objective function to project into the target space, that is, $Q_m \rightarrow R_n$. For the multi-objective task scheduling based on the time limit, energy consumption, and cost in this work, the problem is transformed into an optimization problem for these three mutually influencing objectives. The solution to this problem is a set of Pareto optimal solutions, and a set consisting of each solution represents a task scheduling scheme.

#### 2.1.3. Multi-Objective Task Scheduling Based on Task Publisher

When the task publisher uses the crowdsourcing model to conduct large-scale public remote sensing data collection, the collection is released on a crowdsourcing platform. It is necessary to select different scheduling strategies according to the needs of different remote sensing data to determine the matching between tasks and nodes. Three task scheduling strategies, namely, the time limit priority, energy priority, and cost priority, are common scheduling strategies. Time-limited task scheduling is activated for particularly urgent task assignments. For normalized task assignments without time requirements, energy and cost constraint task scheduling can be utilized. Suppose m collection nodes are $P = \{P_1,\ P_2,\ \ldots P_m\}$; the resources available for each collection node are represented as $P_i = \{Mips_i,\ Energy_i,\ Cost_i\}$, among which $Mips_i$ represents the execution speed of node

$i$ in node set $W$ for a certain type of task, $Energy_i$ represents the energy consumption of node $i$ per unit time, and $Cost_i$ represents the cost of executing unit tasks for node $i$.

The expression $B[i][j]$ represents the matching relationship between task $i(i = 1, 2, \ldots, n)$ and node $j(j = 1, 2, \ldots, m)$, and the assignment of $B[i][j]$ is shown in Equation (2):

$$B[i][j] = \begin{cases} 0 & \text{Task } i \text{ is not assigned to node } j \\ 1 & \text{Task } i \text{ is assigned to node } j \end{cases} \tag{2}$$

In Equation (2), the expression $B[i][j]$ represents the task execution matrix and is defined in Equation (3):

$$B = \begin{bmatrix} B[1][1] & \cdots & B[1][m] \\ \vdots & \ddots & \vdots \\ B[n][1] & \cdots & B[n][m] \end{bmatrix} \tag{3}$$

In Equation (3), each row represents a collection task, and each column represents a node resource. When $\sum_{j=1}^{m} B[i][j] = 1$, it indicates that a task can only be assigned to one node.

Considering the large amount of public remote sensing data, the large data capacity, and the different execution speeds of different collection nodes for different types of tasks, the execution time of each task is determined by the length of the task assigned to the node and the execution speed of the node. For task set $T = \{t_1, t_2, \ldots, t_n\}$, the purpose of dynamic task scheduling in this paper is to find an optimal task scheduling scheme to assign all tasks to the corresponding collection nodes in order to make the total task completion time the shortest. Since the execution speed of different tasks on the node is different, the execution speed of the task on the node can be represented using the execution time matrix in Equation (4):

$$ExecuteTime = \begin{bmatrix} Time_{11} & Time_{12} & \cdots & Time_{1m} \\ Time_{21} & Time_{22} & \cdots & Time_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ Time_{n1} & Time_{n2} & \cdots & Time_{nm} \end{bmatrix} \tag{4}$$

In Equation (4), the symbol $Time_{ij}(1 \le i \le n, 1 \le j \le m)$ represents the execution time of task $i$ on node $j$, and its calculation equation can be expressed in Equation (5):

$$Time_{ij} = \frac{Length_i}{Mips_j} \tag{5}$$

The symbol $Length_i$ represents the task length of task $i$, and $Mips_j$ represents the execution speed of collection node $j$.

When all tasks are distributed and executed, the tasks are executed independently and in parallel on each node. Therefore, the total execution time of the task set is expressed as (6):

$$SumTime = max\{\sum_{i=1}^{n} B[i][j] \times Time[i,j](1 \le j \le m)\} \tag{6}$$

When all tasks in task set $T$ have been executed, if all nodes obtain tasks at the same time, and the total task execution time is determined by the node with the slowest execution speed, the optimization goal of task scheduling is to find an optimal task allocation scheme to minimize $SumTime$.

Due to the differences in the machine configuration of each task execution node, there must be a difference in the energy consumption per unit time when the task is executed. It is known that the power of hardware such as motherboards, processors, graphics cards, hard drives, memory, and monitors is the main factor affecting the energy consumption of

the machine. The total power is the sum of the power of the above hardware. Therefore, when the task scheduling of energy consumption constraints is carried out in this paper, each node is used for a period of time. The energy consumption of running tasks is used as an index to evaluate the energy consumption of each execution node. The total energy consumption of node set $P$ to complete task set $T$ can be expressed as:

$$SumEnergy = \sum_{j=1}^{m} \sum_{i=1}^{n} B[i][j] \times Energy[i][j] \tag{7}$$

in which $Energy[i][j] = Energy_j \times Time[i][j]$, and it represents energy consumption for the total execution time of task set $T_j$ on node set $P_j$. $Energy_j$ represents the energy consumption on node $P_j$ in unit time, that is, the total power of the node. $Time[i][j]$ represents the task execution time of $T_j$ on node $P_j$. The task scheduling of energy consumption constraints aims to find a set of optimal task allocation schemes to minimize $SumEnergy$.

Since each node participating in the collection of public remote sensing data is obtained through crowdsourcing, the task initiator needs to pay remuneration to crowdsourcing users. According to the different abilities of each node to perform tasks, the unit task cost obtained is different. The total cost of the task publisher using node resource set $P$ to complete a batch of tasks $T$ can be expressed as:

$$SumCost = \sum_{j=1}^{m} \sum_{i=1}^{n} B[i][j] \times Cost[i][j] \tag{8}$$

In the equation above, $Cost[i][j] = Cost_j \times count$ represents the total cost of executing task set $T_j$ on node set $P_j$, where $Cost_j$ represents the unit task cost of node $P_j$, and $count$ represents the number of tasks allocated to node $P_j$. The task scheduling of bundles aims to find a set of optimal task assignments that minimize $SumCost$.

From the task issuer's perspective, in this work, when carrying out multi-objective task scheduling of the time limit, energy consumption, and cost, the goal of task scheduling is to map the task set to the appropriate collection node so as to achieve the purpose of realizing the lowest energy consumption and lowest cost. Therefore, the task scheduling objective function can be expressed as:

$$Min(SumTime,\ SumEnergy,\ SumCost) \tag{9}$$

$SumTime$ represents the total task execution time, $SumEnergy$ represents the total energy consumption of the task, and $SumCost$ indicates the total cost of the task.

### *2.2. PSO Optimization Algorithm and PSO-BP Algorithm*

Improved PSO Optimization Algorithm

In order to solve the problem of slow particle optimization speed when the traditional PSO optimization algorithm uses the random initialization of particles, this paper proposes the use of the FCFS, SJF, and MAX_MAX (maximum resources required allocated to maximum capacity) algorithms to initialize the particles, which changes the traditional random initialization method. The specific method is to first use the FCFS, SJF, and MAX_MAX algorithms to perform task scheduling ahead of the PSO optimization algorithm and then execute the PSO optimization algorithm based on the scheduling results.

The pseudo-code of the improved PSO optimization algorithm (Algorithm 1) is as follows:

**Algorithm 1**: Using FCFS algorithm, SJF algorithm, or MAX_MAX algorithm to initialize the particle population.

```
DO
    FOR Particle i
        Use Equation (10) to calculate the fitness value of the particle
    IF The fitness value of the particle at the current position is better than the historical best value of the local particle
        Update the local best individual with the current particles
  END
    IF The current local particle is better than the global best particle of the population
Update the global particles with the current local particle fitness value
    FOR Particle i
        Update the position and velocity of particles according to Formula (12)
  END
WHILE Achieve the maximum number of iterations
```

The advantage of using the above-mentioned improved PSO optimization algorithm is that after applying FCFS, SJF, and MAX_MAX algorithm scheduling, it optimizes the scheduling results of the PSO algorithm, which can speed up the optimization of the entire algorithm. Among them, the fitness function can be expressed as *Fitness*:

$$Fitness = \alpha \cdot Makespan_{position} + (1 - \alpha) \cdot Makespan_{position} \tag{10}$$

In Equation (10), the expression $Makespan_{position}$ represents the total task completion time of the particles under the current allocation plan, which can be expressed as follows:

$$Makespan_{position} = \mathrm{Max}(p_1, p_2, \ldots, p_m) \tag{11}$$

$$\begin{cases} v_i^{k+1} = w \cdot v_i^k + c_1 r_1 \left[ p_i - x_i^k \right] + c_2 r_2 [p_g - x_i^k] \\ x_i^{k+1} = x_i^k + v_i^k, i = 1, 2, \ldots, N \end{cases} \tag{12}$$

The position and velocity of particles in the PSO algorithm are updated using Equation (12). The current iteration number of the particle is represented by $k$, and the inertia weight coefficient is represented by $w$. The larger the value of the inertia weight, the stronger the global search ability of the particle; otherwise, the stronger the local search ability. $c_1$ and $c_2$ are learning factors. $c_1$ describes whether the particle is affected by the individual extreme value, and it enables the particle to have a global search ability to avoid falling into a local solution. $c_2$ represents whether the particle is affected by the global extreme value. $r_1$ and $r_2$ are random numbers in the range (0,1). $w$, $c_1$, and $c_2$ jointly determine the space-searching ability of the particle. The position of the optimal fitness value calculated by the particle in the iterative process is represented by the individual extreme value, which is expressed as $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$, $i = 1, 2, \cdots, N$. The position of the optimal fitness calculated by all particles in the population during the iteration process is represented by the global extreme, which is expressed as $p_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$.

### 2.3. TSCD-TSA Dynamic Task Scheduling Algorithm

Taking into account the differences in the network performance of each task execution node, it has different speeds when performing public remote sensing data collection tasks. In order to realize the prediction of the execution speed of each node task and to provide the basis for the node task execution ability for dynamic task scheduling, TSCD-TSA, combining a node network transmission speed prediction algorithm and a task scheduling algorithm, is proposed. The principle of TSCD-TSA is as follows: Firstly, the BP neural network algorithm is used to assist the timer in dynamically predicting the network transmission capacity of the node, and then the prediction results are used in the dynamic FCFS_PSO

algorithm, dynamic SJF_PSO, dynamic MAX_MAX_PSO, and dynamic PSO algorithms to achieve dynamic task scheduling.

The TSCD-TSA dynamic task scheduling algorithm has the following rules during execution:

- The update frequency of the BP neural network algorithm and dynamic task scheduling algorithm needs to be consistent.
- The BP neural network algorithm is used to adopt a data incremental prediction strategy when predicting; that is, the training data of the algorithm are always taken from the dataset within the most recent period of time to avoid the excessively long training time of the algorithm when the historical collection of data of the node is too large.
- When the dynamic task scheduling algorithm is executed, the scheduling result is updated, and tasks are only reallocated if they have been allocated but the node has not yet started execution. (That is, the result of dynamic task scheduling is the dynamic reallocation of those tasks that have been pre-allocated but not executed.)

## 3. Experiment and Results Analysis

### 3.1. Experimental Environment

CloudSim is a cloud simulation platform launched in 2009 by the University of Melbourne's GRIDS Laboratory [48]. The initial goal of the platform design is to provide a cloud computing platform. Only simulation can complete the control and use of cloud computing service resources, which is convenient for users aiming to complete the deployment of related services and strategies. It is an open-source project developed using the JAVA language. It can run on multiple operating systems, such as Windows and Linux, and users can change its source code or add related functions according to their needs. The platform not only provides the definition of infrastructure in cloud computing but also provides a simulation interface for resource management and task scheduling in cloud computing. The simulation platform mainly has the following core classes: Cloudlet, DataCenter, DataCenterBroker, Host, VirtualMachine, VMScheduler, VMCharacteristics, VMMAllocationPolicy, and VMProvisioner, and the framework provides good interface extension services that can meet task scheduling needs. The version of CloudSim used in this work is 3.0.3. A number of scheduling algorithm classes have been developed under the original framework, including FCFS-PSO, SJF-PSO, Max-Max-PSO, and PSO.

### 3.2. Simulation Experiment

The Sentinel2 acquisition task was used to realize the simulation algorithm experiment of dynamic task scheduling. The task set was taken from one of the four resource nodes, and detailed information about task length and task execution nodes is shown in Table 1. The download power is the average download speed of the node performing this type of task over a period of time. The energy consumption per unit time represents the total power of the node when performing tasks, and the unit task cost represents the node's cost for this type of task.

**Table 1.** List of node resources.

| Node Number | Download Power (kB/s) | Energy Consumption per Unit Time (wh) | Unit Task Cost (CNY) |
|---|---|---|---|
| 1 | 2914.7 | 300 | 2 |
| 2 | 5049.45 | 330 | 4 |
| 3 | 4028.56 | 320 | 3 |
| 4 | 1531.6 | 285 | 1 |

In order to compare the task performance under different algorithms, FCFS-PSO, SJF-PSO, MAX-MAX-PSO, and PSO were each implemented. The maximum iteration number of each algorithm is 500, the number of particles is 20, the number of tasks is 50,

100, and 150, and the number of task nodes is 4. The node network transmission capacity prediction module adopts the BP neural network algorithm, and the update frequency of the prediction algorithm and the task scheduling algorithm is set as 3 min. Table 2 lists the total working hours of each node and the number of tasks assigned to each node for different algorithm scheduling results.

**Table 2.** Task execution results of different algorithms.

| The Number of Tasks | Algorithm | Node 1 | | Node 2 | | Node 3 | | Node 4 | |
|---|---|---|---|---|---|---|---|---|---|
| | | Time (min) | The Number of Tasks | Time (min) | The Number of Tasks | Time (min) | The Number of tasks | Time (min) | The Number of Tasks |
| 50 | FCFS_PSO | 3544 | 14 | 2254 | 18 | 1861 | 14 | 5246 | 4 |
| | SJF_PSO | 3027 | 14 | 1874 | 17 | 2426 | 14 | 5998 | 5 |
| | MAX_MAX_PSO | 3152 | 14 | 1935 | 16 | 2349 | 14 | 5761 | 6 |
| | PSO | 3543 | 13 | 2253 | 13 | 1861 | 12 | 5246 | 12 |
| 100 | FCFS_PSO | 5675 | 33 | 3858 | 31 | 3663 | 23 | 9776 | 13 |
| | SJF_PSO | 5691 | 28 | 3166 | 34 | 4041 | 26 | 11,032 | 12 |
| | MAX_MAX_PSO | 5586 | 30 | 3346 | 33 | 4117 | 22 | 10,439 | 15 |
| | PSO | 5676 | 24 | 3858 | 32 | 3663 | 29 | 9776 | 15 |
| 150 | FCFS_PSO | 8951 | 45 | 6065 | 54 | 5805 | 33 | 14,807 | 18 |
| | SJF_PSO | 8651 | 44 | 5128 | 47 | 6485 | 36 | 16,678 | 23 |
| | MAX_MAX_PSO | 8764 | 37 | 5174 | 50 | 6427 | 42 | 16,464 | 21 |
| | PSO | 8951 | 43 | 6065 | 50 | 5805 | 33 | 14,807 | 24 |

In Table 3, the scheduling results of different algorithms are compared on the basis of the five indexes of the average task execution time, best fitness value, total task execution time, total task energy consumption, and total task cost under different scheduling algorithms.

**Table 3.** Task execution results of different algorithms.

| The Number of Tasks | Algorithm | Average Task Execution Time (min) | Optimum Fitness Value | Total Task Execution Time (min) | Total Task Energy Consumption (Wh) | Total Task Cost (CNY) |
|---|---|---|---|---|---|---|
| 50 | FCFS_PSO | 258 | 2766 | 12,905 | 3,897,650 | 146 |
| | SJF_PSO | 266 | 2768 | 13,324 | 4,012,270 | 143 |
| | MAX_MAX_PSO | 264 | 2657 | 13,196 | 3,977,715 | 140 |
| | PSO | 258 | 2757 | 12,905 | 3,897,020 | 126 |
| 100 | FCFS_PSO | 230 | 4912 | 22,972 | 6,933,960 | 278 |
| | SJF_PSO | 239 | 4876 | 23,930 | 7,189,320 | 282 |
| | MAX_MAX_PSO | 235 | 4653 | 23,488 | 7,072,535 | 273 |
| | PSO | 229 | 5768 | 22,972 | 6,934,260 | 278 |
| 150 | FCFS_PSO | 238 | 9365 | 35,629 | 10,764,345 | 423 |
| | SJF_PSO | 246 | 8397 | 36,943 | 11,115,970 | 407 |
| | MAX_MAX_PSO | 245 | 7844 | 36,830 | 11,085,500 | 421 |
| | PSO | 237 | 8946 | 35,628 | 10,764,345 | 409 |

As shown in Figure 2, Node 4, with the weakest download force, had the least number of tasks assigned but the longest working time. Node 2 and Node 3, with similar differences in download power, basically maintained the same working time during the execution of tasks. Nodes with similar download power had little difference in the number of tasks assigned. Therefore, the download power of nodes is positively correlated with the working time for executing tasks and the number of tasks assigned.

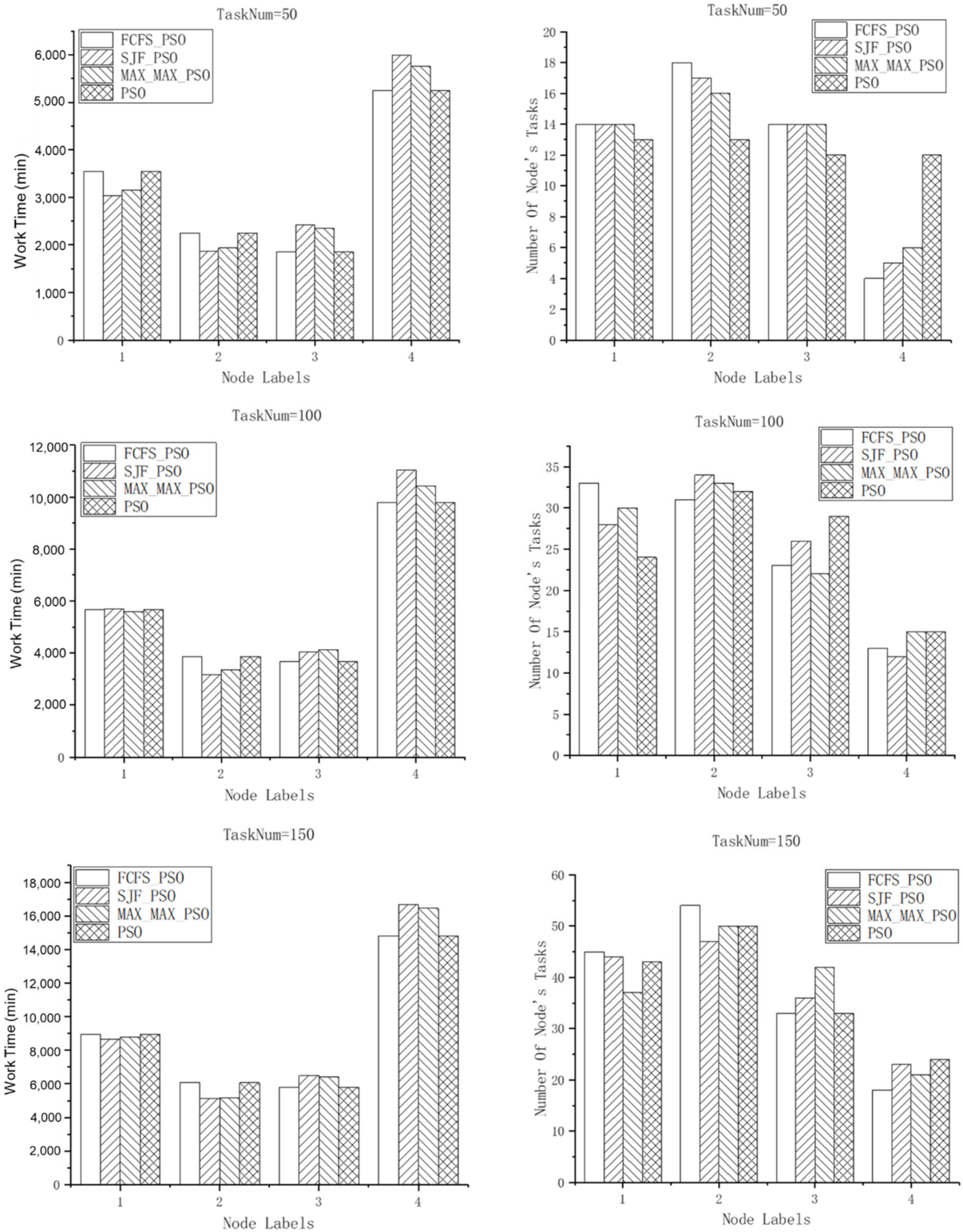


**Figure 2.** The working state of nodes.

Figure 3 shows the optimal fitness value of the particle for different task scheduling results when the task number is 50, 100, and 150. As can be seen in Figure 3, with the increase in the number of tasks, the optimal fitness value of the MAX-MAX-PSO algorithm presents a decreasing trend, which fully indicates that this algorithm is more suitable for the scheduling of a large number of long tasks.

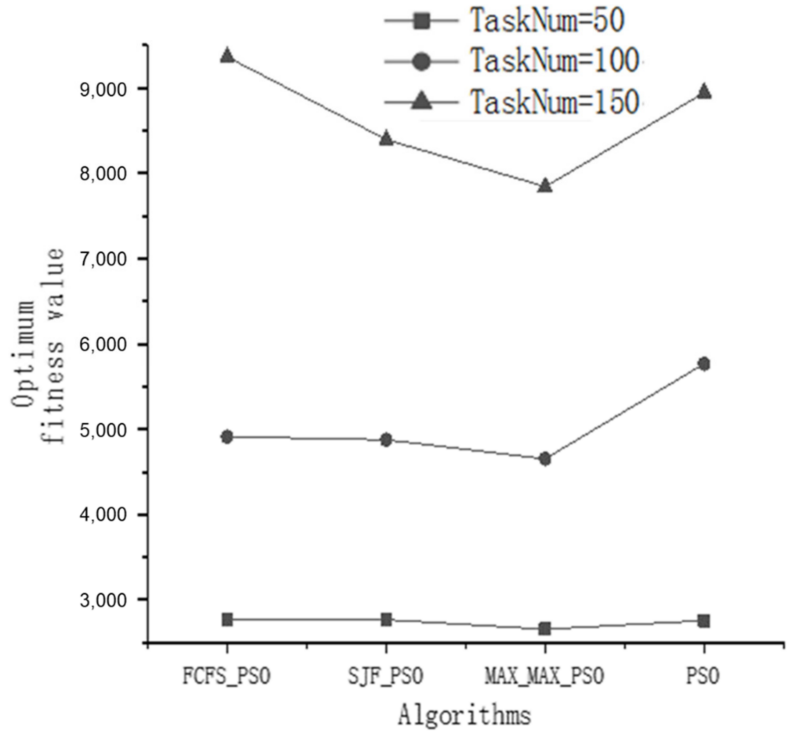


**Figure 3.** Optimal fitness values of different algorithms.

After many experiments and statistical average calculations, Figure 4 presents the convergence curve of four task scheduling algorithms. It can be seen in Figure 4 that the MAX-MAX-PSO algorithm always has the fastest convergence speed.

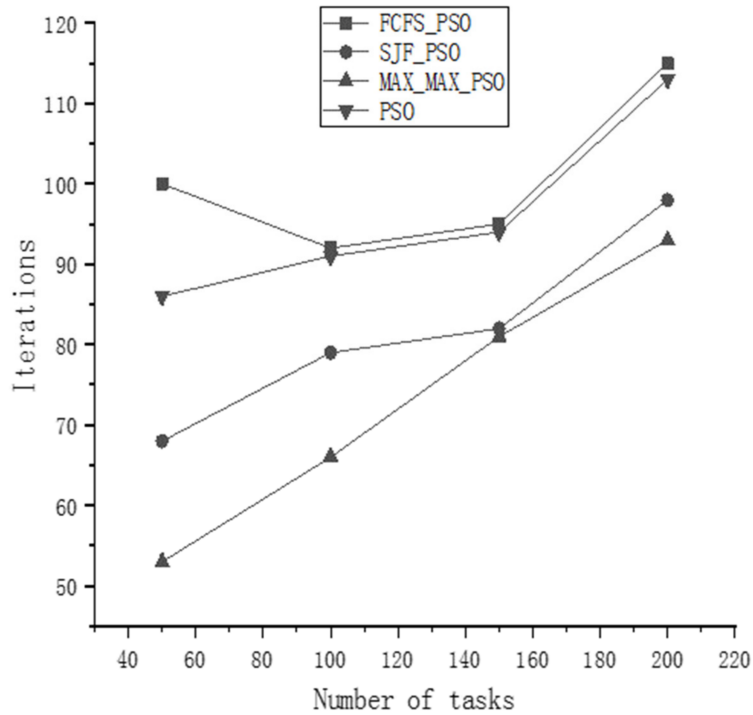


**Figure 4.** Convergence of different algorithms.

## 4. Discussion

Based on the optimization results, MAX-MAX-PSO is the best algorithm, which has the fastest convergence speed and lowest optimal fitness value compared with the convergence speed and optimal fitness value of the other proposed algorithms. In order to make better comparisons between these four different algorithms, the details of the task execution results are presented in Figure 5, where the total task execution time, total task energy consumption, total task cost, average task execution time, average task energy consumption, and average task cost are visualized and compared for three different task numbers: 50, 100, and 150.

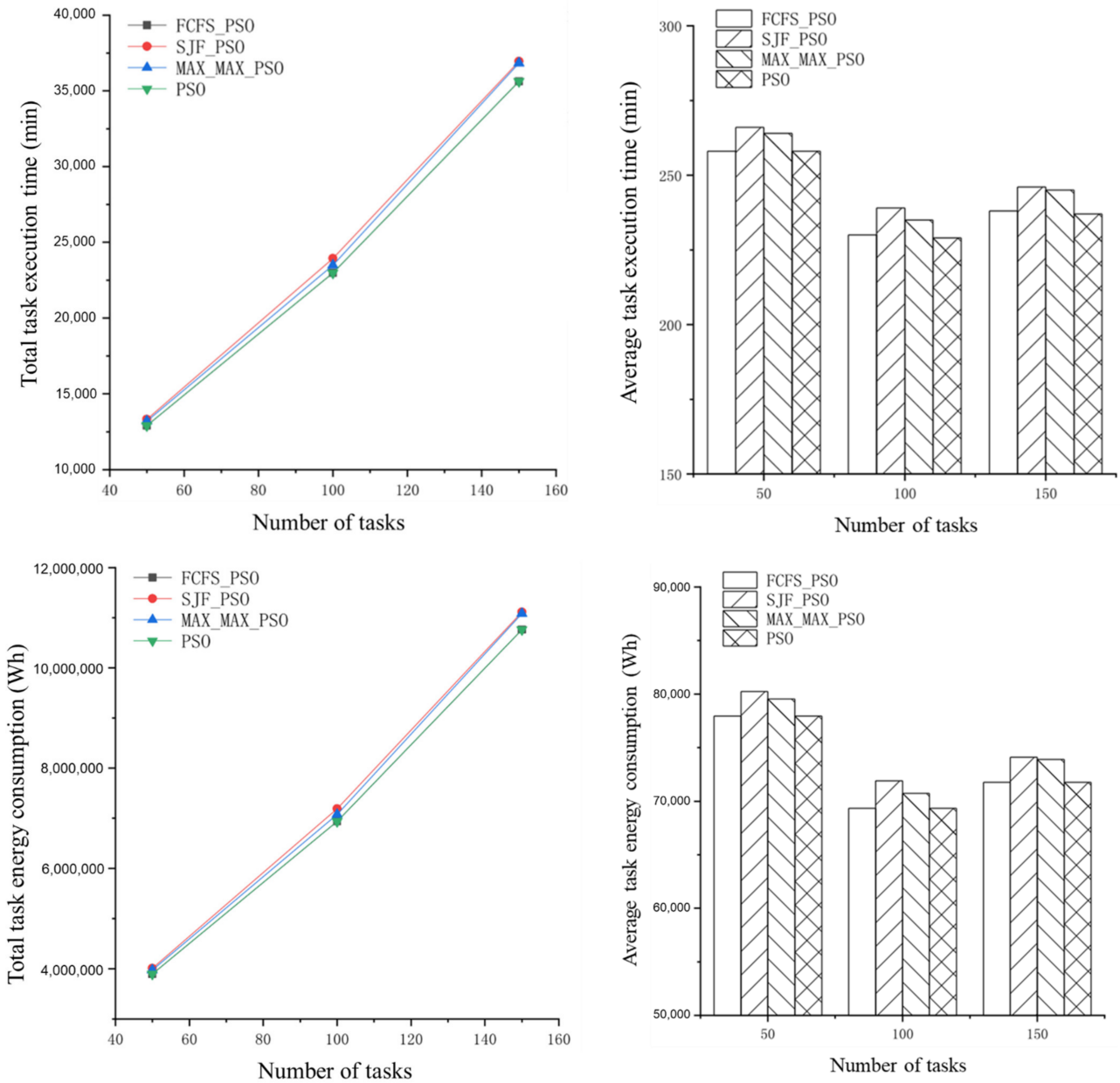


**Figure 5.** *Cont.*

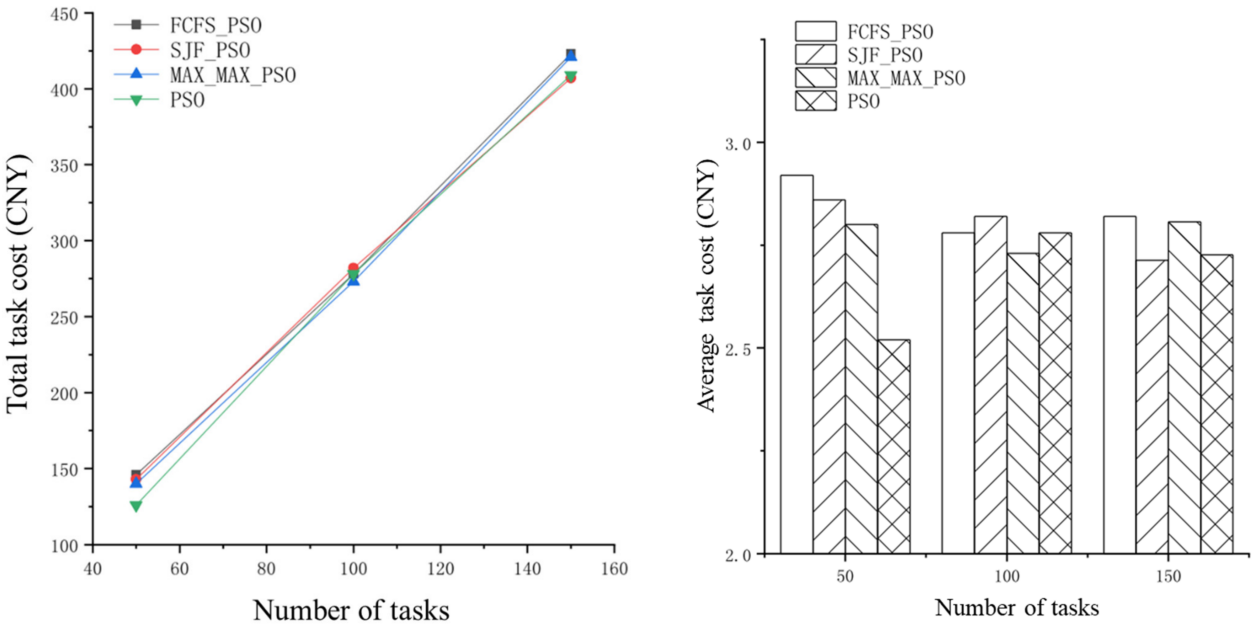


**Figure 5.** Results of task execution of different algorithms.

As shown in Figure 5, when the number of tasks is the same, no matter which task scheduling algorithm is used, the total task execution time, the total energy consumption of the task, and the total cost of the task change very little with the different proposed algorithms. The values of the PSO algorithm are slightly lower than those of FCFS_PSO, SJF_PSO, and MAX_MAX_PSO. In addition, under the condition of the same number of tasks, the average time of the FCFS_PSO algorithm and the PSO algorithm is almost the same. As the number of tasks increases, the average task time, the average energy consumption of tasks, and the average cost of tasks show a decreasing trend.

## 5. Conclusions

This paper first reviews the research on network performance prediction and network-intensive task scheduling. Due to the limitations of the service capabilities of data sources and the constraints of data user node collection capabilities, large-scale public remote sensing data collection has low efficiency and low user collection node utilization. In order to solve these problems, we propose a task scheduling model for open remote sensing data acquisition. An improved PSO-BP algorithm is proposed by improving the inertia weight and learning factor and introducing the dynamic precision adjustment function. In order to address the challenges in the dynamic task allocation of a large amount of remote sensing data, especially to improve the resource utilization of the collection nodes and collection speed, a multi-objective task scheduling model was established. With the consideration of the difference in the speed of the collection task performed by each node at different times, the TSCD-TSA dynamic task scheduling algorithm was developed to improve the traditional PSO optimization algorithm by using FCFS, MAX-MAX, and SJF algorithms. Comparative simulation experiments of four dynamic task scheduling algorithms, namely, FCFS-PSO, SJF-PSO, MAX-MAX-PSO, and PSO, were carried out on the CloudSim platform with the Sentinel2 acquisition task and the BP neural network algorithm as the prediction algorithm of the acquisition node network transmission capacity. The task execution of each node of various task scheduling algorithms was analyzed and compared, and the fitness value and convergence of various algorithms were analyzed. The experimental results show that with the increase in the number of tasks, the fitness value of the MAX-MAX-PSO algorithm presents a decreasing trend, and the convergence speed is obviously accelerated. In the future, we would like to establish a crowdsourced node network transmission capacity prediction model and use deep-learning-based algorithms

to compare with the current algorithms by analyzing the characteristic system of node network transmission capability.

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

**Funding:** This work was supported in part by TUOHAI special project 2020 of the Bohai Rim Energy Research Institute of Northeast Petroleum University under Grant HBHZX202002 and the Project of Excellent and Middle-aged Scientific Research Innovation Team of Northeast Petroleum University under Grant KYCXTD201903.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

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

## References

1. Ma, Y.; Wu, H.; Wang, L.; Huang, B.; Ranjan, R.; Zomaya, A.; Jie, W. Remote sensing big data computing: Challenges and opportunities. *Future Gener. Comput. Syst.* **2015**, *51*, 47–60. [CrossRef]
2. Ma, Y.; Wang, L.; Zomaya, A.Y.; Chen, D.; Ranjan, R. Task-tree based large-scFale mosaicking for massive remote sensed imageries with dynamic dag scheduling. *IEEE Trans. Parallel Distrib. Syst.* **2013**, *25*, 2126–2137. [CrossRef]
3. Wellmann, T.; Lausch, A.; Andersson, E.; Knapp, S.; Cortinovis, C.; Jache, J.; Scheuer, S.; Kremer, P.; Mascarenhas, A.; Kraemer, R. Remote sensing in urban planning: Contributions towards ecologically sound policies? *Landsc. Urban Plan.* **2020**, *204*, 103921. [CrossRef]
4. Zhu, M.; Wang, Z.; Bai, L.; Zhang, J.; Tao, J.; Chen, L. Detection of industrial storage tanks at the city-level from optical satellite remote sensing images. In Proceedings of the Image and Signal Processing for Remote Sensing XXVII, Online, 12 September 2021; SPIE: Bellingham, WA, USA, 2021; Volume 11862, pp. 254–260.
5. Barrett, E.C. *Introduction to Environmental Remote Sensing*; Routledge: London, UK, 2013; ISBN 0203761030.
6. Zhang, J.; Wang, Z.; Bai, L.; Song, G.; Tao, J.; Chen, L. Deforestation Detection Based on U-Net and LSTM in Optical Satellite Remote Sensing Images. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 3753–3756.
7. Wang, Z.; Bai, L.; Song, G.; Zhang, J.; Tao, J.; Mulvenna, M.D.; Bond, R.R.; Chen, L. An oil well dataset derived from satellite-based remote sensing. *Remote Sens.* **2021**, *13*, 1132. [CrossRef]
8. USGS. Available online: https://www.usgs.gov/ (accessed on 5 November 2022).
9. NASA. Available online: https://ladsweb.modaps.eosdis.nasa.gov/search/ (accessed on 5 November 2022).
10. ESA. Available online: https://scihub.copernicus.eu/dhus/#/home (accessed on 5 November 2022).
11. Zhao, Q.; Yu, L.; Li, X.; Peng, D.; Zhang, Y.; Gong, P. Progress and trends in the application of Google Earth and Google Earth Engine. *Remote Sens.* **2021**, *13*, 3778. [CrossRef]
12. Amani, M.; Ghorbanian, A.; Ahmadi, S.A.; Kakooei, M.; Moghimi, A.; Mirmazloumi, S.M.; Moghaddam, S.H.A.; Mahdavi, S.; Ghahremanloo, M.; Parsian, S. Google earth engine cloud computing platform for remote sensing big data applications: A comprehensive review. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *13*, 5326–5350. [CrossRef]
13. Xu, C.; Du, X.; Yan, Z.; Fan, X. ScienceEarth: A Big Data Platform for Remote Sensing Data Processing. *Remote Sens.* **2020**, *12*, 607. [CrossRef]
14. 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]
15. Sarkar, V. Determining average program execution times and their variance. In Proceedings of the ACM SIGPLAN 1989 Conference on Programming Language Design and Implementation, Portland, OR, USA, 19–23 June 1989; pp. 298–312.
16. Soltani, N.; Soleimani, B.; Barekatain, B. Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey. *Int. J. Comput. Netw. Inf. Secur.* **2017**, *9*, 16–22. [CrossRef]
17. Aladwani, T. Types of task scheduling algorithms in cloud computing environment. In *Scheduling Problems: New Applications and Trends*; IntechOpen: London, UK, 2020.
18. Mao, Y.; Chen, X.; Li, X. Max–min task scheduling algorithm for load balance in cloud computing. In *Advances in Intelligent Systems and Computing, Proceedings of the International Conference on Computer Science and Information Technology, Kunming, China, 21–23 September 2013*; Springer: New Delhi, India, 2014; pp. 457–465.

19. Anousha, S.; Ahmadi, M. An improved Min-Min task scheduling algorithm in grid computing. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Grid and Pervasive Computing, Seoul, Korea, 9–11 May 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 103–113.
20. Raj, A.; Kaur, K.; Dutta, U.; Sandeep, V.V.; Rao, S. Enhancement of hadoop clusters with virtualization using the capacity scheduler. In Proceedings of the 2012 Third International Conference on Services in Emerging Markets, Mysore, India, 12–15 December 2012; pp. 50–57.
21. Yadav, R.K.; Mishra, A.K.; Prakash, N.; Sharma, H. An improved round robin scheduling algorithm for CPU scheduling. *Int. J. Comput. Sci. Eng.* **2010**, *2*, 1064–1066.
22. Casanova, H.; Legrand, A.; Zagorodnov, D.; Berman, F. Heuristics for scheduling parameter sweep applications in grid environments. In Proceedings of the Proceedings 9th Heterogeneous Computing Workshop (HCW 2000) (Cat. No. PR00556), Cancun, Mexico, 1 May 2000; pp. 349–363.
23. Xu, X.; Xue, S.; Shi, W. A Heuristic Scheduling Algorithm based on PSO in the Cloud Computing Environment. *Int. J. u-and e-Serv.* **2016**, *9*, 349–362. [CrossRef]
24. Fong, S.; Wong, R.; Vasilakos, A.V. Accelerated PSO Swarm Search Feature Selection for Data Stream Mining Big Data. *IEEE Trans. Serv. Comput.* **2016**, *9*, 33–45. [CrossRef]
25. Hamad, S.A.; Omara, F.A. Genetic-based task scheduling algorithm in cloud computing environment. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 550–556.
26. Fidanova, S.; Durchova, M. Ant algorithm for grid scheduling problem. In *Lecture Notes in Computer Science, Proceedings of the International Conference on Large-Scale Scientific Computing, Sozopol, Bulgaria, 6–10 June 2005*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 405–412.
27. Tripathy, B.; Dash, S.; Padhy, S.K. Dynamic task scheduling using a directed neural network. *J. Parallel Distrib. Comput.* **2015**, *75*, 101–106. [CrossRef]
28. Jena, R.K. Multi objective task scheduling in cloud environment using nested PSO framework. *Procedia Comput. Sci.* **2015**, *57*, 1219–1227. [CrossRef]
29. Kiani, F.; Seyyedabbasi, A.; Nematzadeh, S.; Candan, F.; Çevik, T.; Anka, F.A.; Randazzo, G.; Lanza, S.; Muzirafuti, A. Adaptive Metaheuristic-Based Methods for Autonomous Robot Path Planning: Sustainable Agricultural Applications. *Appl. Sci.* **2022**, *12*, 943. [CrossRef]
30. Elmougy, S.; Sarhan, S.; Joundy, M. A novel hybrid of Shortest job first and round Robin with dynamic variable quantum time task scheduling technique. *J. Cloud Comput.* **2017**, *6*, 12. [CrossRef]
31. Manasrah, A.M.; Ba Ali, H. Workflow Scheduling Using Hybrid GA-PSO Algorithm in Cloud Computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1934784. [CrossRef]
32. 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]
33. Huang, Y.; Chen, Z.; Tao, Y.U.; Huang, X.; Gu, X. Agricultural remote sensing big data: Management and applications. *J. Integr. Agric.* **2018**, *17*, 1915–1931. [CrossRef]
34. Ma, X.; Wang, Z.; Bai, L.; Xu, B.; Gao, J.; Wen, B.; Tao, J. Implementation of a Federated Large-Scale Remote Sensing Data Sharing Platform. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 5771–5774.
35. Guo, J.; Huang, C.; Hou, J. A Scalable Computing Resources System for Remote Sensing Big Data Processing Using GeoPySpark Based on Spark on K8s. *Remote Sens.* **2022**, *14*, 521. [CrossRef]
36. Yu, Z.; Wang, Z.; Bai, L.; Chen, L.; Tao, J. Remote Sensing Inversion of PM10 Based on Spark Platform. In Proceedings of the 2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS, Brussels, Belgium, 11–16 July 2021; pp. 1685–1688.
37. Chebbi, I.; Boulila, W.; Mellouli, N.; Lamolle, M.; Farah, I.R. A comparison of big remote sensing data processing with Hadoop MapReduce and Spark. In Proceedings of the 2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, Tunisia, 21–24 March 2018; pp. 1–4.
38. An, Q.; Hu, Q.; Tang, R.; Rao, L. Intelligent Scheduling Methodology for UAV Swarm Remote Sensing in Distributed Photovoltaic Array Maintenance. *Sensors* **2022**, *22*, 4467. [CrossRef]
39. Wu, X.; Yang, Y.; Sun, Y.; Xie, Y.; Song, X.; Huang, B. Dynamic regional splitting planning of remote sensing satellite swarm using parallel genetic PSO algorithm. *Acta Astronaut.* **2022**, *in press*. [CrossRef]
40. Alkayal, E.S.; Jennings, N.R.; Abulkhair, M.F. Efficient task scheduling multi-objective particle swarm optimization in cloud computing. In Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), Dubai, United Arab Emirates, 7–10 November 2016; pp. 17–24.
41. Gabi, D.; Ismail, A.S.; Zainal, A.; Zakaria, Z.; Al-Khasawneh, A. Cloud scalable multi-objective task scheduling algorithm for cloud computing using cat swarm optimization and simulated annealing. In Proceedings of the 2017 8th International Conference on Information Technology (ICIT), Amman, Jordan, 17–18 May 2017; pp. 1007–1012.
42. Xing, L.; Li, W.; He, M.; Tan, X. Comprehensive multi-objective model to remote sensing data processing task scheduling problem. *Concurr. Comput. Pract. Exp.* **2017**, *29*, e4248. [CrossRef]

43. Chen, H.; Du, C.; Li, J.; Jing, N.; Wang, L. An approach of satellite periodic continuous observation task scheduling based on evolutionary computation. In Proceedings of the Genetic and Evolutionary Computation Conference Companion, Berlin, Germany, 15–19 July 2017; pp. 15–16.
44. Sun, J.; Li, H.; Zhang, Y.; Xu, Y.; Zhu, Y.; Zang, Q.; Wu, Z.; Wei, Z. Multiobjective task scheduling for energy-efficient cloud implementation of hyperspectral image classification. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2020**, *14*, 587–600. [CrossRef]
45. Zhao, Y.; Tian, C.; Fan, J.; Guan, T.; Qiao, C. RPC: Joint Online Reducer Placement and Coflow Bandwidth Scheduling for Clusters. In Proceedings of the International Conference on Network Protocols, ICNP, Cambridge, UK, 25–27 September 2018.
46. Duggan, M.; Shaw, R.; Duggan, J.; Howley, E.; Barrett, E. A multitime-steps-ahead prediction approach for scheduling live migration in cloud data centers. *Softw.-Pract. Exp.* **2019**, *49*, 617–639. [CrossRef]
47. Chao, S.C.; Lin, K.C.J.; Chen, M.S. Flow Classification for Software-Defined Data Centers Using Stream Mining. *IEEE Trans. Serv. Comput.* **2019**, *12*, 105–116. [CrossRef]
48. Calheiros, R.N.; Ranjan, R.; Beloglazov, A.; De Rose, C.A.F.; 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]