electronics

Article

Task Scheduling in Cloud Computing Environment Using Advanced Phasmatodea Population Evolution Algorithms

An-Ning Zhang [1], Shu-Chuan Chu [1], Pei-Cheng Song [1], Hui Wang [2] and Jeng-Shyang Pan [1,3,*]

[1] College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266590, China; zhanganning@sdust.edu.cn (A.-N.Z.); scchu0803@gmail.com (S.-C.C.); spacewe@outlook.com (P.-C.S.)
[2] School of Information Engineering, Nanchang Institute of Technology, Nanchang 330099, China; huiwang@nit.edu.cn
[3] Department of Information Management, Chaoyang University of Technology, No. 168 Jifeng E. Rd., Wufeng District, Taichung 413310, Taiwan
* Correspondence: jengshyangpan@gmail.com

Abstract: Cloud computing seems to be the result of advancements in distributed computing, parallel computing, and network computing. The management and allocation of cloud resources have emerged as a central research direction. An intelligent resource allocation system can significantly minimize the costs and wasting of resources. In this paper, we present a task scheduling technique based on the advanced Phasmatodea Population Evolution (APPE) algorithm in a heterogeneous cloud environment. The algorithm accelerates up the time taken for finding solutions by improving the convergent evolution of the nearest optimal solutions. It then adds a restart strategy to prevent the algorithm from entering local optimization and balance its exploration and development capabilities. Furthermore, the evaluation function is meant to find the best solutions by considering the makespan, resource cost, and load balancing degree. The results of the APPE algorithm being tested on 30 benchmark functions show that it outperforms similar algorithms. Simultaneously, the algorithm solves the task scheduling problem in the cloud computing environment. This method has a faster convergence time and greater resource usage when compared to other algorithms.

Keywords: cloud computing; Phasmatodea Population Evolution algorithm; task scheduling; heterogeneous

check for updates

Citation: Zhang, A.-N.; Chu, S.-C.; Song, P.-C.; Wang, H.; Pan, J.-S. Task Scheduling in Cloud Computing Environment Using Advanced Phasmatodea Population Evolution Algorithms. *Electronics* **2022**, *11*, 1451. https://doi.org/10.3390/electronics11091451

Academic Editor: Claus Pahl

Received: 9 March 2022
Accepted: 26 April 2022
Published: 30 April 2022

1. Introduction

Cloud computing has received attention as an innovative model due to the fast growth of Internet technology [1]. In the cloud computing system, distributed computing technology and various open service interfaces are used to obtain benefits by selling its redundant computing and storage capabilities to users [2], i.e., the cloud computing concept is based on a pay-per-use paradigm that consists of a general populace set of the on-demand assignment of programmable computer resources, and users can easily access the network [3]. Service providers can make profits by providing essential services during a short period of time through the advantages of their hardware resources, while users and certain companies that do not want to increase the cost of their internal data center construction can lease virtualized resources provided by service providers to save costs [4].

The goal of cloud computing is to offer consumers cloud services using virtualized resources [5]. Generally speaking, each service provided by cloud service providers can represent a task, and the process of service providers leasing services to different cloud users is the process of task allocation and execution [6]. Due to the significantly increased demand for cloud services, cloud service providers are extending the range and quantity of services that they offer [7]. In recent years, the emphasis has switched to how to correctly and appropriately plan these operations in order to enhance resource usage. For these scheduling problems, it is actually a kind of NP-hard problem [8]. Researchers have put forth much effort on this subject. For example, Panda et al. [9] devised the assessment criteria while keeping time and resource usage in mind and provided the scheduling method. In a diverse context, Hussain et al. [10] suggested an energy-saving and effective job scheduling technique. In addition, due to the simple operation, scalability, and robustness, heuristic algorithms are effective at addressing issues such as task scheduling, load balancing, and virtual machine placement, and have gained a lot of attention from researchers.

The Phasmatodea Population Evolution (PPE) algorithm is a heuristic population evolution algorithm based on stick insect population development features [11]. The algorithm treats the problem's solutions as a population, and each population has two properties: the population size and pace of expansion [12]. The movement of the solution of the problem in the solution space is regarded as the evolutionary trend of the population [13]. In addition, the technique substitutes K-nearest optimal solutions for optimal global solutions. This broadens the range of population evolution patterns and the speed at which the best solutions are found is increased, whilst the population competition influences the evolutionary direction of the population [14]. In the context of this research, we present an APPE algorithm that optimizes and balances the algorithm's development and exploration skills to improve the algorithm's optimization speed and general search capacity [15]. The contributions of this paper are as follows:

- It proposes an advanced Phasmatodea Population Evolution (APPE) algorithm. It combines the restart mechanism with a new evolutionary trend of stick insect populations to balance the algorithm's exploration and development capabilities.
- It builds an evaluation function using the makespan, cost, and load balancing degree as indicators.
- Extensive simulation and comparison of the proposed approach with comparable algorithms utilizing CEC2014 benchmark suites for testing.
- To assess the performance of the algorithm, it is compared to five similar algorithms in two heterogeneous environments.

The following is the architecture of this essay, the second part introduces the related work on cloud computing task scheduling, the third part introduces the proposed algorithm, the fourth section introduces the simulation strategy and the result analysis, and finally, the fifth section introduces the summary and analyses.

2. Related Work

Cloud computing has grown in popularity as a distributed system based on the Internet in recent years. It has attracted much attention because it can configure a shareable resource pool according to the request and can be immediately provided and unloaded [16]. As the number of cloud users continues to grow, the amount of requests to be processed in the cloud is also increasing. If the job scheduling is unreasonable, its performance will be reduced [17]. The scheduling algorithm is responsible for distributing the work requested by cloud users to cloud resources in order to minimize the make span, improve resource utilization, reduce use cost, and balance the cloud infrastructure and resource load [18–20]. The scheduling question in the cloud environment can be separated into different fields, which are illustrated as follows.

Task scheduling algorithms can be divided into task scheduling based on the Quality of Service (Qos), task scheduling based on the Ant Colony Optimization (ACO) algorithm, Particle Swarm Optimization (PSO) algorithm-based task scheduling, Genetic Algorithm (GA) based on the task scheduling, and fuzzy-based task scheduling [21].

Initially, individuals researched task scheduling focused on QoS. He et al. [22] was a pioneer in the field of QoS-based scheduling algorithm research, taking QoS parameters as evaluation criteria to analyze and execute the next schedule. The suggested approach is a generic adjustable heuristic task scheduling algorithm based on QoS advice that significantly improves performance by incorporating QoS into the Min-Min heuristic algorithm.

Following that, Wu et al. [23] devised a cloud-based QoS-driven job scheduling algorithm which prioritizes the tasks according to the particularity of the tasks, allocates resources according to the sorting queue, and allocates resources according to the sorting queue. It was tested and found to achieve good load balancing and performance. Through the suggested optimization model and distributed queue based on dynamic load, the S et al. [24] enhances resource usage efficiency, minimizes cost, and delays. HGEDH et al. [25] categorized the tasks based on their attributes and assigned them to the available servers. However, the algorithm can only categorize tasks based on the supplied attributes, and tasks must be queued for categorization. Hanini et al. [26] utilized the workload to determine the number of virtual machines to be used, combined with the control of incoming requests to virtual machines to control energy consumption.

ACO also plays an important role in addressing scheduling problems. Ants release pheromones in the process of searching for food, and share their travel experience through the trail of pheromones to connect with each other. ACO is effective for solving NP-complete problems, especially for dynamic task scheduling problems [27]. Liu et al. [28] applies the ACO algorithm to solve the problem of virtual machine placement in the cloud while also decreasing energy consumption by reducing the number of physical hosts. Xin et al. [29] introduced the network and offered an ACO-based resource scheduling strategy for reducing the running time and energy consumption by improving the scheduling system. Delavar et al. [30] studied the task scheduling problem of grid computing in terms of QoS. Researchers used a sub-heuristic ACO strategy in terms of make span and money. However, the task scheduling problem in a heterogeneous environment is not considered. Wu et al. researched the assessment model under multi-objective in order to monitor the change of energy consumption at all times, which considerably decreased the resource waste [31]. Recently, Kumar et al. [32] utilized the scheduler to arrange work and resources in a reasonable manner, coupled the ACO and GA algorithms, and devised a multi-objective scheduling system. Nevertheless, the load and security of virtual machines were not taken into account in the scheduling procedure in this study. To make it more efficient and balance the load in the cloud, Ragmani et al. [33] considered the allocation of the execution time and resources, and optimized the efficiency of the algorithm by altering parameters. Sun et al. [34] found the optimal solutions earlier by optimizing the updating method of the ant colony pheromone, and proposed a Period Ant Colony Optimization (PACO) algorithm.

Because of its simple principle and less parameters to be adjusted, using the PSO algorithm to address scheduling problems has garnered a lot of interest. In the scheduling process, Pandey et al. [35] incorporated the transport and execution costs into the evaluation function and solved the scheduling issue using the PSO algorithm. The algorithm was then compared to the Best Resource Selection (BRS) algorithm, which saves a significant amount of money. Juan et al. [36] improved the PSO algorithm which defined the cost vector to restrict the initial solutions and the search space of the solutions. Alsaidy et al. [37] stood for an advanced optimization algorithm that utilized heuristics to initialize particle swarms in order to maximize cost-effectiveness and resource consumption. Wen et al. [38] mixed the ACO and PSO algorithms to accelerate the exit from local optimization and enhanced the convergence speed. Kumar et al. [39] offered a task selection strategy for improving the algorithm's performance by picking the optimal VM.

Several methods for dealing with redundant task scheduling problems have been proposed based on studies of improved GA. Kumar et al. [40] integrated the GA algorithm with the techniques typically used in work scheduling, proposed a new algorithm, and compared it with the standard genetic algorithm. Nevertheless, the research only considered the makespan under different resource quantities when evaluating. Nagar et al. [41] used a previous workflow scheduling model that predicted the earliest completion time, and reduced the execution time by proposing a new Predict Earliest Finish Time (PEFT) genetic algorithm. However, the technique is best suited for workflows with a small number of tasks and does not consider the execution cost, the number of virtual machines, or the data center's energy usage. Velliangiri et al. [42] combined the hybrid electric search with the genetic algorithm while considering the execution time, maximum completion time, and load balancing. Similarly, the article did not consider the load of virtual machines. Manasrah et al. [43] introduced the GA-PSO algorithm to handle the resource allocation problem in the cloud, substantially shortening the make span and cost. Farhadian et al. [44] optimized the virtual machine allocation problem by using a hybrid algorithm based on IC and GA, thereby reducing the energy consumption of the data center, and the simulation experiments on CloudSim software obtained good results.

Fahmy [45] explained the use of fuzzy algorithms to schedule aperiodic work in real-time systems for fuzzy-based scheduling. On a soft real-time single processor system, Fahmy developed a fuzzy method for aperiodic work scheduling. The total throughput time of the task is lowered by assessing the priority of the work that is now running and altering the priority of the job in the queue, and the technique is utilized in a multi-objective algorithm [46]. Zhou et al. [47] proposed a heterogeneous earliest completion time algorithm based on fuzzy dominance sorting in the Infrastructure as a Service (IaaS) workflow, which greatly improved the running speed. Revathi et al. [48] combined the scheduling optimization with virtual allocation methods to design a VM based on security requirements, and proposed a scheduling heuristic optimization algorithm based on the Cost Prediction Matrix (CPM). Rezaeipanah et al. [49] devised a mechanism for managing virtual machines in the cloud center.

Based on the meta heuristic algorithm and cloud computing task scheduling, combined with the above research results and problems, this research focused on using the APPE algorithm to solve the task scheduling problem in the cloud. The simulation results on MATLAB show that the APPE algorithm can get a better allocation scheme in a heterogeneous cloud environment.

3. Task Scheduling Based on the Advanced Phasmatodea Population Evolution Algorithms

In cloud computing, scheduling can be divided into task scheduling and job scheduling. Among them, Hadoop is the most often used scheduling method in task scheduling [50,51], which is mainly used to improve the system performance. Task scheduling is used to allocate resources to task applications by mapping tasks and resources, and now the critical issue in task scheduling is determining how to assign jobs to processors to achieve low cost, high efficiency, and minimize the makespan [52]. A new method is proposed here to address the shortcomings of the current research on cloud computing task scheduling algorithms. First, a task scheduling system and evaluation model are designed to evaluate the pros and cons of task allocation based on the load level, cost, and makespan [53]. The APPE-based task scheduling method is then implemented to the task scheduling model. Based on the evaluation framework, the optimal work scheduling method is obtained.

3.1. System Design

When a cloud user submits a job, the task is placed to the task queue via the task manager, and the task scheduling method assigns the task to the virtual machine. Each task is independent and non-preemptive. Figure 1 depicts the cloud computing system's task scheduling paradigm.

Tasks are serially processed on the virtual machine in this article via the task queue. The task has two attributes: m and len, m represents the amount of tasks. len represents the length of the task, and the unit is millions of machine language instructions (MI). A virtual machine has four properties: n, $MIPS$, RAM, $bandwidth$. n represents the amount of VMs, and $MIPS$ represents the average processing rate of single-word fixed-point instructions. RAM stands for memory, $bandwidth$ stands for bandwidth.

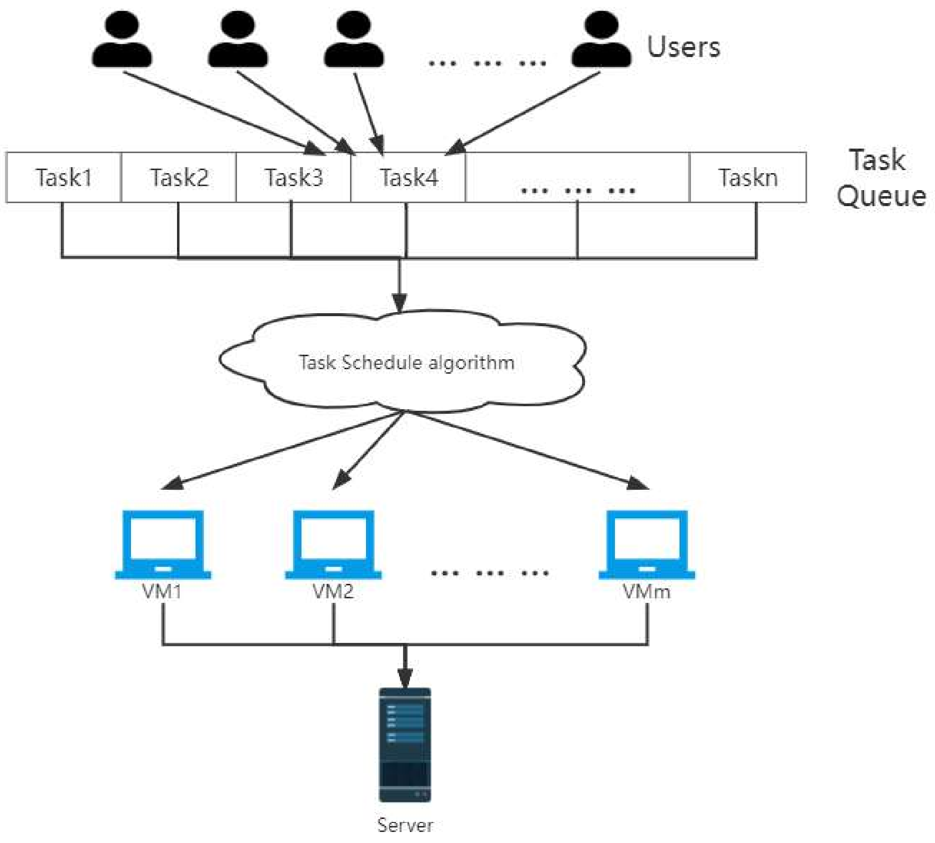


Figure 1. Cloud computing task scheduling model.

3.2. Evaluation Model

The indicators for evaluating cloud computing task scheduling performance mainly include makespan, Profit, Completion time, Cost, and Waiting time. This paper comprehensively considers the task scheduling performance from the perspectives of makespan, cost and degree of imbalance [54]. These performance indicators are described in detail below.

Let $Task = \{T_1, T_2, \ldots, T_m\}, (50 \leq m \leq 500)$. *Task* represents the task queue submitted by cloud users, and m stands for the number of tasks. *Let* $T\_length = \{len_1, len_2, \ldots, len_m\}$. len_i stands for the length of the i-th task. $VM = \{VM_1, VM_2, \ldots, VM_n\}$. VM_j represents the j-th VM. n stands for the amount of VM. $ESC = \{ESC_{ij}\}_{m*n}$, $ESC_{ij} = 1$ represents the fact that the task i is executed on the VM j, otherwise $ESC_{ij} = 0$. $ETC = \{ETC_{ij}\}_{m*n}$ stands for the expected completion time, that is, the processing time of the task i on the VM j, which is computed by the following formula:

$$ETC_{ij} = \frac{len_i}{MIPS_j} \tag{1}$$

$MIPS_j$ represents the executing speed of the VM j.

- Makespan:
  Makespan is a critical metric for assessing the effectiveness of task scheduling in the cloud. The makespan is the completion time of the task, which reflects the total operating duration of all VMs, and is computed using the following formula:

$$Makespan = \max_j(\sum_{i=1}^{m} ETC_{ij} * ESC_{ij}) \tag{2}$$

- Cost:

The cost calculated according to the specification of VM is as follows; $USD0.12$, $USD0.13$, $USD0.17$, $USD0.48$, $USD0.52$, and $USD0.96$ per hour [55]. Calculate the cost of the virtual machine through the following formula.

$$Cost = \sum_{j=1}^{n}(cost_j * (\sum_{i=1}^{m} ETC_{ij} * ESC_{ij})) \tag{3}$$

$cost_j$ represents the hourly cost of the jth virtual machine, and in a heterogeneous environment, its resource cost is related to $MIPS$, RAM, and $bandwidth$, RAM stands for virtual machine memory, $bandwidth$ represents the bandwidth of VM.

- Load:

$$Load = \sqrt{\varphi * \frac{\sum_{j=1}^{n} load_j * VL_j}{n * Makespan}} \tag{4}$$

φ represents the degree of imbalance of the system. n represents the number of VM, $load_j$ represents the degree of impact of each MIPS, RAM and bandwidth on the virtual machine, as shown by the following formula:

$$\varphi = \sqrt{\frac{\sum_{j=1}^{n}(VL_j - \overline{VL_j})^2}{n}} \tag{5}$$

$$load_j = \zeta * MIPS + \delta * RAM + \eta * bandwidth \tag{6}$$

$$VL_j = \sum_{i=1}^{m} ETC_{ij} * ESC_{ij} \tag{7}$$

$$\overline{VL_j} = \frac{\sum_{j=1}^{n} VL_j}{n} \tag{8}$$

Here, VL_j stands for the running time of the VM i, $\overline{VL_j}$ represents the average running time of the VM, $load_j$ is related to MIPS, RAM and bandwidth, ζ, δ, η are three weight values, respectively. Through the above performance indicators, the objective function formula is obtained as follows:

$$fitness = Makespan * Cost * Load \tag{9}$$

3.3. Scheduling Model Based on the APPE

3.3.1. Advanced PPE Algorithm

In the development process of Phasmatodea populations, the PPE algorithm guaranteed the development of continuous populations by mimicking the qualities of route dependency, convergence evolution, population expansion, and competitiveness. The evolution process can be thought of as an optimization process of populations in d-dimensional space; hence, we can think of the solution as a population of stick insects in d-dimensional space.

As a new heuristic algorithm, PPE has good optimization and exploration ability. The algorithm considers the top K solutions with the highest fitness value to be the closest to optimum solutions. It improves its global exploration ability through the K-nearest optimal solutions and perturbation. Through path dependence, convergent evolution, population growth, and competition, the solutions other than the nearest optimal solutions can reach the region with a better fitness value faster and improve their optimization ability.

In our work, the advanced PPE algorithm is divided into two aspects. On the one hand, the speed searching for the ideal solution is increased by refining the population evolution trend calculation formula. As we are all aware, the calculation of the population's evolution trend ev is the basis of the original PPE method, which affects the exploration ability of the population. The updating technique is divided into three sections. The first component is the closest ideal population, while the second component represents the continuation of the prior evolution trend, and the third part is mutation [56]. The calculation formula of the evolution trend is as follows:

$$ev^{k+1} = (1 - p^{k+1}) * A + p^{k+1} * (ev^k + m) \tag{10}$$

In the formula, ev stands for the evolutionary trend of the population, p represents the population quantity, A denotes the level of similarity to the nearest optimum, m stands for mutation, and k is the amount of iterations.

Through the formula, it was found that the first part of the convergent evolution has no effect on the update of the evolution trend of the nearest optimal population, so we propose a new evolution trend update formula, as shown below.

$$ev^{k+1} = (1 - p^{k+1}) * A + p^{k+1} * (ev^k + m) + rand * B * flag \tag{11}$$

In the improved formula, $flag$ is used to judge whether it is the nearest optimal solution, and B denotes the degree to which it resembles the global optimum.

Another aspect of improving the PPE algorithm is to add a restart mechanism to the algorithm, restarting the solutions with poor fitness values and the worst solutions among the nearest optimal solutions and avoiding the area centered on the initial position of the restart solutions during the restart process [57]. Algorithm 1 depicts the pseudo-code for the advanced PPE algorithm.

Algorithm 1 Advanced PPE Algorithm

```
Initialize $N$ populations, $T$;
Initialize $ev$, $P$, $K$;
Calculate $f(T)$, set $nbest$ and $gbest$;
while $t <$ Iter do
    Update $T$ to $newT$;
    Calculate $f(newT)$, update $nbest$ and $gbest$;
    Restart ordinary solutions;
    Restart nearest optimal solutions;
    while $i = 1 < N$ do
        if $f(newT) \geq f(T)$ then
            if $Rand < P_i$ then
                $T = newT$;
                Update $f(T)$, $P_i$;
            end if
            Update $ev_i$ use Equation (11);
        else
            $T = newT$;
            Update $f(T)$, $P_i$;
            Update $ev_i$ use Equation (11);
        end if
        Solution for random choice $T_j$, ($j \neq i$);
        if $dist(x_j, x_i) < G$ then
            Update $ev_i$, $P_i$;
        end if
    end while
end while
```

In Algorithm 1, N is the number of Phasmatodea populations, T stands for the set of Phasmatodea populations in the APPE algorithm, and each population represents a solutions. Furthermore, $gbest$ denotes the global optimum solutions, whereas $nbest$ denotes the nearest optimal solutions.

After the calculation of the fitness values of all solutions in each iteration process is completed, a restart mechanism is introduced to restart the ordinary solutions and the nearest optimal solutions, and the population evolution trend after restart is updated. The restart process of the two types of solutions is the same as in Algorithm 2, which is as follows.

Algorithm 2 Restart Strategy

```
Initialize $G$, $G_{max}$, $t$, $K$, $iter$;//the value of iter is related to restart count
if $G>G_{max}$ || rem($t$, $iter$)==0 then
    Restart $k$ solutions with poor fitness;
    Avoid the position where the solutions start;
    Modify population evolution trends;
    Calculate fitness after restarting the solutions update;
end if
```

3.3.2. Task Scheduling Algorithm Based on the Advanced PPE

Since task scheduling seeks to assign n independent non-preemptive tasks to cloud resources (VMs). In this work, we map each solution to a population; each stick insect population symbolizes a task scheduling technique; the dimension reflects the amount of tasks; and the value of the dimension i of the solutions is j, which means that task i is allocated to the VM j. Taking the proposed evaluation function as the standard, we obtain the optimal resource allocation scheme based on the APPE algorithm [58,59]. The execution procedure is depicted in Algorithm 3 below.

Algorithm 3 APPE based task scheduling

```
Step1: Initialize the parameter, generate initial population $N_t$;
Step2: According to the proposed evaluation function, calculate fitness $F(N_t)$;
Step3: The initial global optimal and nearest optimal solutions are obtained;
Step4: The evolutionary trend of the population is calculated through path dependence,
convergent evolution, population growth and competition;
Step5: Update $newN_t$ and calculate new fitness $F(newN_t)$;
Step6: Update $N_t$, global optimal and nearest optimal solutions;
Step7: Repeat steps 4, 5, and 6 until the iteration is complete;
Step8: The global optimal solution is the optimal scheduling scheme;
```

In Algorithm 3, N_t represents the position of solutions population and each solution represents the location of a stick insect population, that is, a task-scheduling scheme. $newN_t$ represents the position of the updated solutions population.

4. Experiments

In this section, the simulation experiments and result analysis are mainly introduced, consisting of two parts. First, extensive simulation experiments are performed on the APPE algorithm using the CEC2014 [60] benchmark suites. The algorithm has several test functions, however, the CEC competition contains the generally used standard test suite. To tackle the scheduling problem, we employed the APPE algorithm. The Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Gravitational Search Algorithm (GSA), Butterfly Optimization Algorithm (BOA), and PPE algorithms are introduced in the simulation experiments to compare with the APPE algorithm to more explicitly demonstrate the algorithm's effectiveness.

4.1. CEC 2014 Benchmark Function Test

The CEC2014 benchmark suites contain 30 benchmark functions, including unimodal, multimodal, mixed and composite functions. To promote equality in the comparison, all methods in this study are assessed 30,000 times, the number of populations is 30, and the dimension is 30. Run each algorithm 30 times on various benchmark functions to determine the mean value and compare the results [61]. This is shown in Table 1.

Table 1. Comparison with the average fitness functions of PSO, BOA, GA, GSA, and PPE.

$F(x)$	PSO	BOA	GSA	GA	PPE	APPE
F_1	1.9144×10^7 (<)	2.0540×10^9 (<)	3.7244×10^7 (<)	1.3810×10^9 (<)	1.5112×10^8 (<)	9.1968×10^6
F_2	1.2384×10^4 (>)	8.6593×10^{10} (<)	1.0383×10^4 (>)	7.0194×10^{10} (<)	1.0129×10^9 (<)	1.8361×10^5
F_3	8.7023×10^3 (<)	2.5869×10^5 (<)	7.1666×10^4 (<)	1.5172×10^6 (<)	4.5206×10^4 (<)	1.7352×10^3
F_4	6.0384×10^2 (<)	2.1446×10^4 (<)	8.3904×10^2 (<)	1.0381×10^4 (<)	8.1387×10^2 (<)	5.2926×10^2
F_5	5.2092×10^2 (<)	5.2136×10^2 (<)	5.2000×10^2 (>)	5.2080×10^2 (<)	5.2102×10^2 (<)	5.2004×10^2
F_6	6.1649×10^2 (>)	6.4896×10^2 (<)	6.3146×10^2 (<)	6.4134×10^2 (<)	6.3256×10^2 (<)	6.2694×10^2
F_7	7.0099×10^2 (>)	1.5987×10^3 (<)	9.7041×10^2 (<)	1.3437×10^3 (<)	7.5837×10^2 (<)	7.0101×10^2
F_8	8.5619×10^2 (>)	1.2821×10^3 (<)	9.4964×10^2 (<)	1.2125×10^3 (<)	1.0379×10^3 (<)	9.1413×10^2
F_9	9.6858×10^2 (>)	1.5070×10^3 (<)	1.0612×10^3 (<)	1.2635×10^3 (<)	1.1685×10^3 (<)	1.0477×10^3
F_{10}	1.9757×10^3 (>)	1.0350×10^4 (<)	4.7221×10^3 (<)	8.9101×10^3 (<)	6.3289×10^3 (<)	2.6770×10^3
F_{11}	4.8385×10^3 (<)	1.0541×10^4 (<)	5.3969×10^3 (<)	9.3226×10^3 (<)	6.9845×10^3 (<)	4.7614×10^3
F_{12}	1.2023×10^3 (<)	1.2059×10^3 (<)	1.2000×10^3 (>)	1.2031×10^3 (<)	1.2025×10^3 (<)	1.2004×10^3
F_{13}	1.3005×10^3 (<)	1.3099×10^3 (<)	1.3004×10^3 (<)	1.3073×10^3 (<)	1.3006×10^3 (<)	1.3004×10^3
F_{14}	1.4003×10^3 (<)	1.7676×10^3 (<)	1.4003×10^3 (<)	1.6209×10^3 (<)	1.4009×10^3 (<)	1.4003×10^3
F_{15}	1.5138×10^3 (>)	8.1727×10^5 (<)	1.5131×10^3 (<)	2.9158×10^5 (<)	2.3954×10^3 (<)	1.5207×10^3
F_{16}	1.6127×10^3 (<)	1.6143×10^3 (<)	1.6137×10^3 (<)	1.6136×10^3 (<)	1.6128×10^3 (<)	1.6124×10^3
F_{17}	1.5357×10^6 (<)	3.0978×10^8 (<)	1.3658×10^6 (<)	1.3668×10^8 (<)	5.2233×10^6 (<)	6.7243×10^5
F_{18}	4.2728×10^5 (<)	7.6174×10^9 (<)	2.4175×10^3 (>)	4.0280×10^9 (<)	4.9473×10^6 (<)	3.7621×10^3
F_{19}	1.9119×10^3 (>)	2.6985×10^3 (<)	2.0044×10^3 (<)	2.3272×10^3 (<)	1.9734×10^3 (<)	1.9118×10^3
F_{20}	7.5211×10^3 (<)	1.7481×10^6 (<)	7.9748×10^4 (<)	1.0676×10^6 (<)	4.1029×10^4 (<)	4.2456×10^3
F_{21}	4.5395×10^5 (<)	1.4404×10^8 (<)	2.8147×10^5 (<)	6.3294×10^7 (<)	1.3997×10^6 (<)	2.2188×10^5
F_{22}	2.5628×10^3 (>)	8.3690×10^4 (<)	3.2168×10^3 (<)	8.5066×10^3 (<)	2.9189×10^3 (<)	2.7239×10^3
F_{23}	2.6165×10^3 (<)	3.7753×10^3 (<)	2.6164×10^3 (<)	2.5350×10^3 (<)	2.6561×10^3 (<)	2.5019×10^3
F_{24}	2.6388×10^3 (<)	2.7454×10^3 (<)	2.6081×10^3 (<)	2.6032×10^3 (<)	2.6563×10^3 (<)	2.6019×10^3
F_{25}	2.7117×10^3 (<)	2.7847×10^3 (<)	2.7050×10^3 (<)	2.7006×10^3 (<)	2.7248×10^3 (<)	2.7000×10^3
F_{26}	2.7138×10^3 (<)	2.8593×10^3 (<)	2.8001×10^3 (<)	2.7975×10^3 (<)	2.7881×10^3 (<)	2.7004×10^3
F_{27}	3.3299×10^3 (<)	4.9037×10^3 (<)	4.8355×10^3 (<)	2.9759×10^3 (<)	3.4583×10^3 (<)	2.9220×10^3
F_{28}	4.0983×10^3 (<)	1.1954×10^4 (<)	6.3250×10^3 (<)	3.0835×10^3 (<)	7.9688×10^3 (<)	3.0019×10^3
F_{29}	8.0862×10^6 (<)	9.2385×10^8 (<)	2.0767×10^4 (<)	4.1516×10^7 (<)	1.2057×10^6 (>)	4.5678×10^6
F_{30}	1.0400×10^4 (<)	1.6330×10^7 (<)	1.0514×10^5 (<)	2.7623×10^6 (<)	9.8454×10^4 (<)	9.2265×10^3
< / = / >	21 / 0 / 9	30 / 0 / 0	26 / 0 / 4	30 / 0 / 0	29 / 0 / 1	

For Table 1, the symbol $(<)$ indicates that the approach which outperforms the current benchmark function which is inferior to that of the APPE algorithm, and the symbol $(>)$ signifies that the technique outperforms the APPE algorithm on the present benchmark function. Finally, the symbol $(=)$ stands for the effectiveness of the method on the current benchmark function, which is similar to that of PPE. The comparison results on each function are shown at the end of Table 1. The APPE algorithm outperforms the PSO, BOA, GA, GSA, and PPE algorithms in terms of total performance among the 30 benchmark functions, among which APPE outperforms BOA and GA on all test functions, only one benchmark function is worse than the PPE algorithm, and 26 results are better than GSA. Compared with PSO, only 21 benchmark functions are better than PSO, and among the remaining 9 benchmark functions, 2 benchmark functions are unimodal functions while the remaining seven are mixing functions. This demonstrates that the optimization ability of the PSO algorithm is better in specific functions, but the exploration ability is generally worse than that of APPE. On the whole, because of the randomization of the algorithm assessment, the APPE algorithm performs better [62].

When the relevance threshold is set to 0.05, the Wilcoxon's sign rank test results for PSO, BOA, GA, GSA, and PPE algorithm and the APPE algorithm are shown in Table 2. Among them, the indication $(+)$ means that the APPE algorithm performance is superior when the significance level is 0.05 under the current function, and the indication $(-)$ denotes that the APPE algorithm performance is worse than that of the algorithm.

Table 2. Comparison with the results of PSO, BOA, GA, GSA and PPE at a significant level α = 0.05 under the Wilcoxon's signed rank test.

$F(x)$	PSO	BOA	GSA	GA	PPE
F_1	4.5153×10^{-4} (+)	1.5099×10^{-11} (+)	2.3080×10^{-10} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_2	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_3	7.7904×10^{-9} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_4	1.0141×10^{-7} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_5	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_6	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	5.7833×10^{-8} (+)	1.5099×10^{-11} (+)	4.4455×10^{-10} (+)
F_7	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_8	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	3.2591×10^{-9} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_9	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	4.6670×10^{-2} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{10}	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{11}	7.8118×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	2.0420×10^{-5} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{12}	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.0000×10^{0} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{13}	3.8693×10^{-6} (+)	1.5099×10^{-11} (+)	1.1129×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	1.9101×10^{-10} (+)
F_{14}	9.2874×10^{-4} (+)	1.5099×10^{-11} (+)	2.0177×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	1.6692×10^{-11} (+)
F_{15}	9.9938×10^{-1} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{16}	1.3770×10^{-3} (+)	1.5080×10^{-11} (+)	3.6901×10^{-11} (+)	1.3034×10^{-10} (+)	1.4194×10^{-4} (+)
F_{17}	4.0723×10^{-5} (+)	1.5090×10^{-11} (+)	3.1405×10^{-6} (+)	1.5099×10^{-11} (+)	2.4876×10^{-11} (+)
F_{18}	2.3195×10^{-5} (+)	1.5099×10^{-11} (+)	9.9218×10^{-1} $(-)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{19}	9.9602×10^{-1} $(-)$	1.5099×10^{-11} (+)	2.7470×10^{-11} (+)	1.5099×10^{-11} (+)	8.4736×10^{-10} (+)
F_{20}	2.7806×10^{-4} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{21}	1.6643×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	7.4724×10^{-2} $(\approx)$	1.5099×10^{-11} (+)	1.9101×10^{-10} (+)
F_{22}	9.9902×10^{-1} $(-)$	1.5099×10^{-11} (+)	6.0116×10^{-9} (+)	1.5099×10^{-11} (+)	2.2296×10^{-4} (+)
F_{23}	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	3.3825×10^{-5} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{24}	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	6.7929×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{25}	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.3543×10^{-2} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{26}	2.4909×10^{-4} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	2.4876×10^{-11} (+)
F_{27}	5.4683×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	5.3328×10^{-8} (+)	1.5099×10^{-11} (+)
F_{28}	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)
F_{29}	9.9999×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	9.0463×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	1.0000×10^{0} $(-)$
F_{30}	5.3240×10^{-1} $(\approx)$	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)	1.5099×10^{-11} (+)

Table 2 shows that the APPE algorithm outperforms the PSO, BOA, GSA, GA, and PPE algorithms in terms of overall performance. Finally, the Friedman test is shown in Table 3.

Table 3. Friedman test.

Function	Sum of Squares	Degree of Freedom	Mean Squares	p-Value
F_1	4.8847×10^{2}	5	9.7693×10^{1}	2.2171×10^{-28}
F_2	5.0833×10^{2}	5	1.0167×10^{2}	1.3767×10^{-29}
F_3	4.9640×10^{2}	5	9.9280×10^{1}	7.3103×10^{-29}
F_4	5.0313×10^{2}	5	1.0063×10^{2}	2.8500×10^{-29}
F_5	5.0047×10^{2}	5	1.0009×10^{2}	4.1388×10^{-29}
F_6	4.9533×10^{2}	5	9.9067×10^{1}	8.4866×10^{-29}
F_7	5.2127×10^{2}	5	1.0425×10^{2}	2.5197×10^{-26}
F_8	5.1380×10^{2}	5	1.0276×10^{2}	6.4053×10^{-30}
F_9	5.1107×10^{2}	5	1.0221×10^{2}	9.3906×10^{-30}
F_{10}	5.1613×10^{2}	5	1.0323×10^{2}	4.6205×10^{-30}
F_{11}	4.6787×10^{2}	5	9.3573×10^{1}	3.9463×10^{-27}
F_{12}	4.8173×10^{2}	5	9.6347×10^{1}	5.6836×10^{-28}

Table 3. *Cont.*

Function	Sum of Squares	Degree of Freedom	Mean Squares	p-Value
F_{13}	4.6647×10^{2}	5	9.3293×10^{1}	4.7988×10^{-27}
F_{14}	4.7707×10^{2}	5	9.5413×10^{1}	1.0912×10^{-27}
F_{15}	5.0693×10^{2}	5	1.0139×10^{2}	1.6746×10^{-29}
F_{16}	4.4293×10^{2}	5	8.8587×10^{1}	1.2823×10^{-25}
F_{17}	4.8733×10^{2}	5	9.7467×10^{1}	2.5978×10^{-28}
F_{18}	4.8947×10^{2}	5	9.7893×10^{1}	1.9278×10^{-28}
F_{19}	5.0260×10^{2}	5	1.0052×10^{2}	3.0708×10^{-29}
F_{20}	4.8827×10^{2}	5	9.7653×10^{1}	2.2800×10^{-28}
F_{21}	4.3720×10^{2}	5	8.7440×10^{1}	2.8532×10^{-25}
F_{22}	4.7253×10^{2}	5	9.4507×10^{1}	2.0564×10^{-27}
F_{23}	4.4827×10^{2}	5	8.9653×10^{1}	6.0920×10^{-26}
F_{24}	4.7887×10^{2}	5	9.5773×10^{1}	8.4849×10^{-28}
F_{25}	4.5980×10^{2}	5	9.1960×10^{1}	1.2176×10^{-26}
F_{26}	4.5840×10^{2}	5	9.1680×10^{1}	1.4805×10^{-26}
F_{27}	4.8987×10^{2}	5	9.7973×10^{1}	1.8229×10^{-28}
F_{28}	5.2127×10^{2}	5	1.0425×10^{2}	2.2520×10^{-30}
F_{29}	3.6307×10^{2}	5	7.2613×10^{1}	8.6256×10^{-21}
F_{30}	4.9567×10^{2}	5	9.8933×10^{1}	9.3160×10^{-29}

4.2. Heterogeneous Cloud Environment Test

After an extensive simulation of the APPE algorithm using the CEC2014 benchmark suite, we put the APPE algorithm through its paces in a heterogeneous cloud environment and compared the evaluation functions with the PSO, BOA, GA, GSA, and PPE algorithm. Again, to guarantee the equality of the comparison, set the number of populations to 30 and the number of iterations to 1000. At the same time, we create two separate heterogeneous settings to assess the algorithm's performance. In the first environment (represented by Environment 1 in the future), the MIPS of the VM is different, and the other parameters are the same. In the second environment (represented by Environment 2 in the future), the MIPS, RAM and bandwidth of the VM are different, and the values are all random numbers within a specific range. The relevant parameters of each heterogeneous environment are shown in Tables 4 and 5.

Table 4. Environment 1 parameters.

Entity	Parameter	Values
Task	Nm of Task	50–500
	Length	100–1000
Virtual Machine	Nm of VM	15
	RAM	512 MB
	MIPS	100–1000
	Bandwidth	1000 MB
	Size	10,000
	VVM	XUN
	Operating-System	Linux
	Nm of CPUs	1
Host	Nm of Host	2
	RAM	2048 MB
	Storage	1,000,000
	Bandwidth	10,000
Data Center	Amount	2

Table 5. Environment 2 parameters.

Entity	Parameter	Values
Task	Nm of Task	50–500
	Length	100–1000
Virtual Machine	Nm of VM	25
	RAM	128–15,360 MB
	MIPS	256–30,720
	Bandwidth	128–15,360 MB
	Size	10 GB
	VVM	XUN
	Operating-System	Linux
	Nm of CPUs	1
Host	Nm of Host	2
	RAM	20 GB
	Storage	1 TB
	Bandwidth	10 GB
Data Center	Amount	2

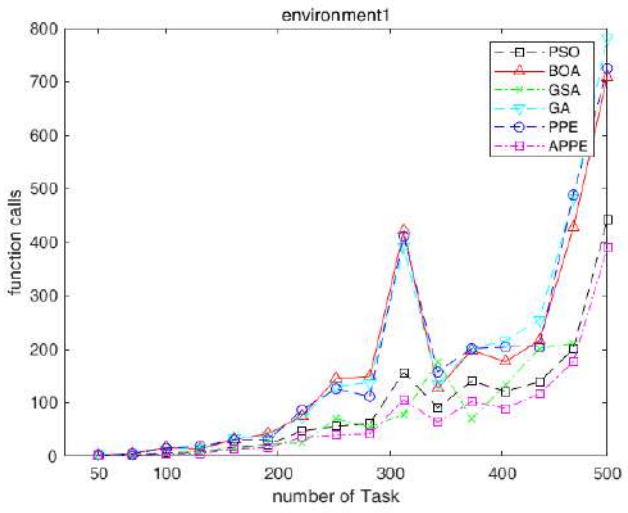
We put the algorithm through its paces in two heterogeneous environments. The method is compared to similar algorithms under various numbers of work conditions to demonstrate the algorithm's stability and efficiency. Figure 2 shows the algorithm performance in the two heterogeneous environments with the number of tasks ranging from 50 to 500, where the x axis stands for the amount of tasks and the y axis denotes the performance index evaluation function value.

In Figure 2a, the overall performance of the PPE, BOA, and GA algorithm is the worst. The evaluation results of the PSO algorithm are similar to that of the APPE algorithm, but the effect is always worse than that of APPE algorithm when the number of tasks is different. In the meantime, the GSA algorithm occasionally has better evaluation results than the APPE algorithm, but on the whole, the APPE algorithm is more robust. Because the random characteristic is introduced into the GSA algorithm, the force of the agent in the D dimension is the random weighted sum of the forces exerted on it by other agents in this dimension, which makes the GSA algorithm less robust.

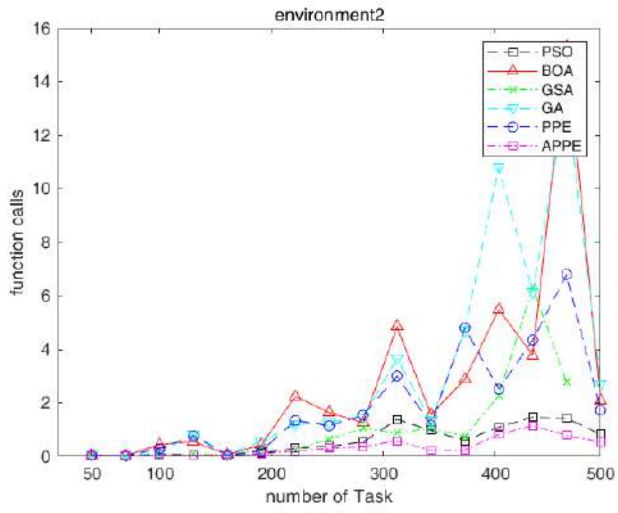
In Figure 2b, with the increasing number of tasks, the performance of PPE, BOA, BA and GSA algorithms gradually deteriorates. The PSO algorithm and APPE algorithm always maintain this good performance, but the performance of the APPE algorithm is better.

In order to properly show the algorithm's convergence, Figures 3 and 4 show the convergence of the algorithm with a different amount of tasks in two heterogeneous environments. Due to space constraints, this article only displays a subset of the results. Only the scenarios when the number of jobs is little, big, or in the medium range are presented for each heterogeneous environment.

In Figures 3 and 4, when the APPE algorithm is compared to similar algorithms, it is discovered that the APPE algorithm has a quicker convergence speed and a lower adaption value and significant advantages in different tasks and different heterogeneous environments. In short, the application of the APPE algorithm has a good performance and effect in solving the task scheduling issues.

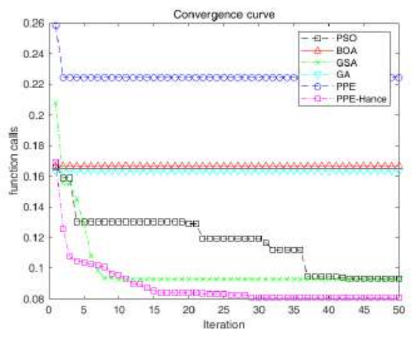


(**a**) Environment 1

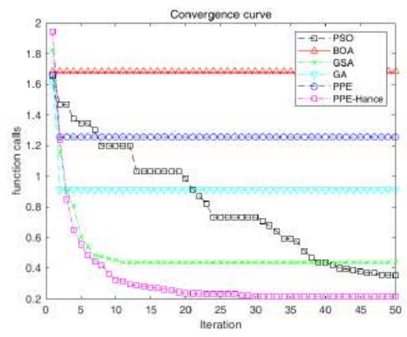


(**b**) Environment 2

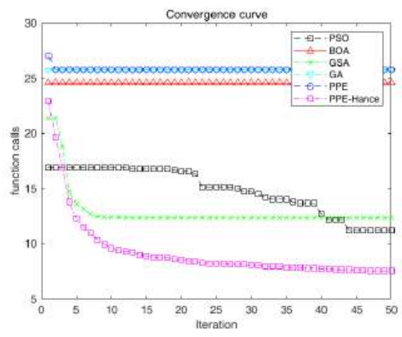
Figure 2. Performance comparison of algorithms in a heterogeneous environment.



(**a**) T_num=55

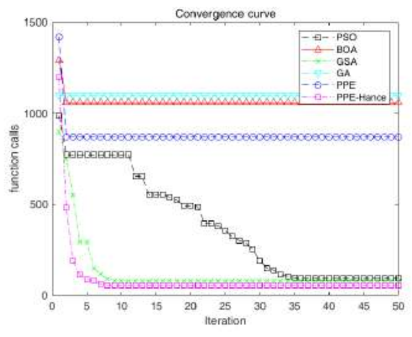


(**b**) T_num=147

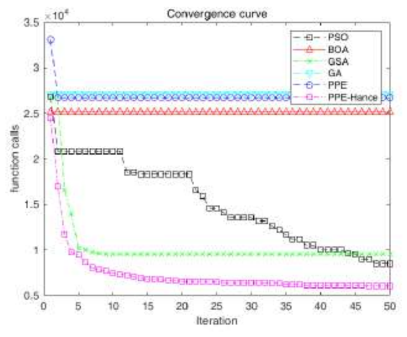


(**c**) T_num=485

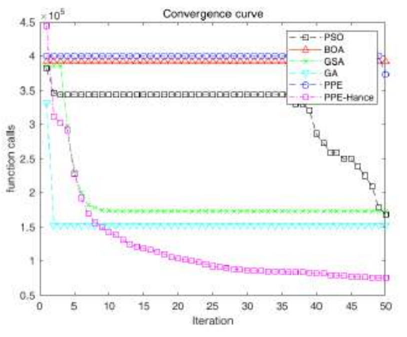
Figure 3. The convergence comparison of different numbers of tasks in heterogeneous Environment 1.



(**a**) T_num=57



(**b**) T_num=148



(**c**) T_num=484

Figure 4. The convergence comparison of different numbers of tasks in heterogeneous Environment 2.

5. Conclusions

An advanced PPE algorithm was suggested throughout this research to address the task scheduling problem in a heterogeneous cloud environment. By optimizing a new heuristic algorithm PPE, it balances the optimization and exploration capabilities of the solutions. The provided task scheduling performance indicators are used to thoroughly analyze the influence of numerous elements on the ultimate rental cost. However, the scheduling model proposed in this research is only suitable for solving the static task scheduling problems, and cannot allocate resources at any time according to the arrival time of tasks which will be solved in future work. The experiments revealed that the advanced PPE method is particularly successful at solving NP-hard problems, lowering the cost and improving the resource usage of virtual machines. Our future work is to study multi-objective problems. At the same time, we wanted to explore the dynamic load balancing mechanism to alleviate the problem of the cloud resource waste.

Author Contributions: Conceptualization, A.-N.Z. and S.-C.C.; formal analysis, J.-S.P., A.-N.Z., P.-C.S. and S.-C.C.; methodology, J.-S.P., A.-N.Z., S.-C.C. and H.W.; validation, J.-S.P., H.W. and P.-C.S.; writing—original draft preparation, A.-N.Z.; and writing—review and editing, J.-S.P., A.-N.Z., S.-C.C., P.-C.S. and H.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

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

References

1. Panda, S.K.; Jana, P.K. Efficient task scheduling algorithms for heterogeneous multi-cloud environment. *J. Supercomput.* **2015**, *71*, 1505–1533. [CrossRef]
2. Yiqiu, F.; Xia, X.; Junwei, G. Cloud Computing Task Scheduling Algorithm Based On Improved Genetic Algorithm. In Proceedings of the 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 15–17 March 2019; pp. 852–856. [CrossRef]
3. Mesbahi, M.R.; Hashemi, M.; Rahmani, A.M. Performance evaluation and analysis of load balancing algorithms in cloud computing environments. In Proceedings of the 2016 Second International Conference on Web Research (ICWR), Tehran, Iran, 27–28 April 2016; pp. 145–151. [CrossRef]
4. Lin, W.; Liang, C.; Wang, J.Z.; Buyya, R. Bandwidth-aware divisible task scheduling for cloud computing. *Softw. Pract. Exp.* **2014**, *44*, 163–174. [CrossRef]
5. Garg, S.K.; Versteeg, S.; Buyya, R. A framework for ranking of cloud computing services. *Future Gener. Comput. Syst.* **2013**, *29*, 1012–1023. [CrossRef]
6. Tsai, J.T.; Fang, J.C.; Chou, J.H. Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Comput. Oper. Res.* **2013**, *40*, 3045–3055. [CrossRef]
7. Agarwal, D.; Jain, S. Efficient optimal algorithm of task scheduling in cloud computing environment. *arXiv* **2014**, arXiv:1404.2076.
8. Dordaie, N.; Navimipour, N.J. A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments. *ICT Express* **2018**, *4*, 199–202. [CrossRef]
9. Panda, S.K.; Jana, P.K. An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems. *Clust. Comput.* **2019**, *22*, 509–527. [CrossRef]
10. Hussain, M.; Wei, L.F.; Lakhan, A.; Wali, S.; Ali, S.; Hussain, A. Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing. *Sustain. Comput. Inform. Syst.* **2021**, *30*, 100517. [CrossRef]
11. Zhou, J.L.; Chu, S.C.; Peng, Y.J.; Huang, K.C.; Pan, J.S. An advanced clustering algorithm based on k-means and phasmatodea population evolution algorithm. *Data Sci. Pattern Recognit.* **2020**, *4*, 41–56.
12. Song, P.C.; Chu, S.C.; Pan, J.S.; Yang, H. Phasmatodea population evolution algorithm and its application in length-changeable incremental extreme learning machine. In Proceedings of the 2020 2nd International Conference on Industrial Artificial Intelligence (IAI), Shenyang, China, 23–25 October 2020; pp. 1–5. [CrossRef]
13. Pan, J.S.; Song, P.C.; Pan, C.A.; Abraham, A. The Phasmatodea Population Evolution Algorithm and Its Application in 5G Heterogeneous Network Downlink Power Allocation Problem. *J. Internet Technol.* **2021**, *22*, 1199–1213.
14. Zhu, Y.; Yan, F.; Pan, J.S.; Yu, L.; Bai, Y.; Wang, W.; He, C.; Shi, Z. Mutigroup-Based Phasmatodea Population Evolution Algorithm with Mutistrategy for IoT Electric Bus Scheduling. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 1500646. [CrossRef]
15. Pan, J.S.; Liu, N.; Chu, S.C. A Hybrid Differential Evolution Algorithm and Its Application in Unmanned Combat Aerial Vehicle Path Planning. *IEEE Access* **2020**, *8*, 17691–17712. [CrossRef]
16. Keshanchi, B.; Souri, A.; Navimipour, N.J. An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: Formal verification, simulation, and statistical testing. *J. Syst. Softw.* **2017**, *124*, 1–21. [CrossRef]
17. Liu, C.Y.; Zou, C.M.; Wu, P. A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing. In Proceedings of the 2014 13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science, Xi'an, China, 24–27 November 2014; pp. 68–72. [CrossRef]
18. Alhaidari, F.; Balharith, T.; AL-Yahyan, E. Comparative Analysis for Task Scheduling Algorithms on Cloud Computing. In Proceedings of the 2019 International Conference on Computer and Information Sciences (ICCIS), Sakaka, Saudi Arabia, 3–4 April 2019; pp. 1–6. [CrossRef]

19. Halim, A.H.A.; Hajamydeen, A.I. Cloud Computing Based Task Scheduling Management Using Task Grouping for Balancing. In Proceedings of the 2019 IEEE 9th International Conference on System Engineering and Technology (ICSET), Shah Alam, Malaysia, 7 October 2019; pp. 419–424. [CrossRef]
20. Geng, X.; Yu, L.; Bao, J.; Fu, G. A task scheduling algorithm based on priority list and task duplication in cloud computing environment. In *Web Intelligence*; IOS Press: Amsterdam, The Netherlands, 2019; Volume 17, pp. 121–129.
21. Arunarani, A.; Manjula, D.; Sugumaran, V. Task scheduling techniques in cloud computing: A literature survey. *Future Gener. Comput. Syst.* **2019**, *91*, 407–415. [CrossRef]
22. He, X.; Sun, X.; Von Laszewski, G. QoS guided min-min heuristic for grid task scheduling. *J. Comput. Sci. Technol.* **2003**, *18*, 442–451. [CrossRef]
23. Wu, X.; Deng, M.; Zhang, R.; Zeng, B.; Zhou, S. A Task Scheduling Algorithm based on QoS-Driven in Cloud Computing. *Procedia Comput. Sci.* **2013**, *17*, 1162–1169. [CrossRef]
24. Potluri, S.; Rao, K.S. Optimization model for QoS based task scheduling in cloud computing environment. *Indones. J. Electr. Eng. Comput.* **2020**, *18*, 1081–1088. [CrossRef]
25. Ali, H.G.E.D.H.; Saroit, I.A.; Kotb, A.M. Grouped tasks scheduling algorithm based on QoS in cloud computing network. *Egypt. Inform. J.* **2017**, *18*, 11–19.
26. Hanini, M.; Kafhali, S.E.; Salah, K. Dynamic VM allocation and traffic control to manage QoS and energy consumption in cloud computing environment. *Int. J. Comput. Appl. Technol.* **2019**, *60*, 307–316. [CrossRef]
27. Gupta, A.; Garg, R. Load Balancing Based Task Scheduling with ACO in Cloud Computing. In Proceedings of the 2017 International Conference on Computer and Applications (ICCA), Doha, Qatar, 6–7 September 2017; pp. 174–179. [CrossRef]
28. Liu, X.F.; Zhan, Z.H.; Du, K.J.; Chen, W.N. Energy aware virtual machine placement scheduling in cloud computing based on ant colony optimization approach. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Vancouver, BC, Canada, 12–16 July 2014; pp. 41–48.
29. Xin, G. Ant colony optimization computing resource allocation algorithm based on cloud computing environment. In Proceedings of the International Conference on Education, Management, Computer and Society, Shenyang, China, 1–3 January 2016.
30. Delavar, A.G.; Bayrampoor, J.; Boroujeni, A.R.K.; Broumandnia, A. Task scheduling in grid environment with ant colony method for cost and time. *Int. J. Comput. Sci. Eng. Appl.* **2012**, *2*, 1. [CrossRef]
31. Wu, J.; Xu, M.; Liu, F.F.; Huang, M.; Ma, L.; Lu, Z.M. Solar Wireless Sensor Network Routing Algorithm Based on Multi-Objective Particle Swarm Optimization. *J. Inf. Hiding Multimed. Signal Process.* **2021**, *12*, 1–11.
32. Senthil Kumar, A.; Venkatesan, M. Multi-objective task scheduling using hybrid genetic-ant colony optimization algorithm in cloud environment. *Wirel. Pers. Commun.* **2019**, *107*, 1835–1848. [CrossRef]
33. Ragmani, A.; Elomri, A.; Abghour, N.; Moussaid, K.; Rida, M. FACO: A hybrid fuzzy ant colony optimization algorithm for virtual machine scheduling in high-performance cloud computing. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 3975–3987. [CrossRef]
34. Sun, W.; Zhang, N.; Wang, H.; Yin, W.; Qiu, T. PACO: A Period ACO Based Scheduling Algorithm in Cloud Computing. In Proceedings of the 2013 International Conference on Cloud Computing and Big Data, Fuzhou, China, 16–19 December 2013; pp. 482–486. [CrossRef]
35. 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 2010 24th IEEE International Conference on Advanced Information Networking and Applications, Perth, WA, Australia, 20–23 April 2010; pp. 400–407. [CrossRef]
36. Juan, W.; Fei, L.; Aidong, C. An improved PSO based task scheduling algorithm for cloud storage system. *Adv. Inf. Sci. Serv. Sci.* **2012**, *4*, 465–471.
37. Alsaidy, S.A.; Abbood, A.D.; Sahib, M.A. *Heuristic Initialization of PSO Task Scheduling Algorithm in Cloud Computing*; Computer and Information Sciences, Journal of King Saud University: Riyadh, Saudi Arabia, 2020. [CrossRef]
38. Wen, X.; Huang, M.; Shi, J. Study on Resources Scheduling Based on ACO Allgorithm and PSO Algorithm in Cloud Computing. In Proceedings of the 2012 11th International Symposium on Distributed Computing and Applications to Business, Engineering Science, Guilin, China, 19–22 October 2012; pp. 219–222. [CrossRef]
39. Kumar, M.; Sharma, S.C. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing. *Neural Comput. Appl.* **2020**, *32*, 12103–12126. [CrossRef]
40. Kumar, P.; Verma, A. Independent task scheduling in cloud computing by improved genetic algorithm. *Int. J. Adv. Res. Comput. Sci. Softw. Eng.* **2012**, *2*, 5.
41. Nagar, R.; Gupta, D.K.; Singh, R.M. Time effective workflow scheduling using genetic algorithm in cloud computing. *Int. J. Inf. Technol. Comput.* **2018**, *10*, 68–75. [CrossRef]
42. Velliangiri, S.; Karthikeyan, P.; Arul Xavier, V.; Baswaraj, D. Hybrid electro search with genetic algorithm for task scheduling in cloud computing. *Ain Shams Eng. J.* **2021**, *12*, 631–639. [CrossRef]
43. Manasrah, A.M.; Ba Ali, H. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing. *Wirel. Commun. Mob. Comput.* **2018**, *2018*, 1934784. [CrossRef]
44. Farhadian, F.; Kashani, M.M.R.; Rezazadeh, J.; Farahbakhsh, R.; Sandrasegaran, K. WITHDRAWN: An efficient IoT cloud energy consumption based on genetic algorithm. *Digit. Commun. Netw.* **2019**. [CrossRef]

45. Fahmy, M. A fuzzy algorithm for scheduling non-periodic jobs on soft real-time single processor system. *Ain Shams Eng. J.* **2010**, *1*, 31–38. [CrossRef]
46. Lv, L.; Zhou, X.D.; Kang, P.; Fu, X.F.; Tian, X.M. Multi-Objective Firefly Algorithm with Hierarchical Learning. *J. Netw. Intell.* **2021**, *6*, 411–427.
47. Zhou, X.; Zhang, G.; Sun, J.; Zhou, J.; Wei, T.; Hu, S. Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. *Future Gener. Comput. Syst.* **2019**, *93*, 278–289. [CrossRef]
48. Sujana, J.A.J.; Revathi, T.; Rajanayagam, S.J. Fuzzy-based Security-Driven Optimistic Scheduling of Scientific Workflows in Cloud Computing. *IETE J. Res.* **2020**, *66*, 224–241. [CrossRef]
49. Rezaeipanah, A.; Mojarad, M.; Fakhari, A. Providing a new approach to increase fault tolerance in cloud computing using fuzzy logic. *Int. J. Comput. Appl.* **2022**, *44*, 139–147. [CrossRef]
50. Xu, X.; Cao, L.; Wang, X. Adaptive Task Scheduling Strategy Based on Dynamic Workload Adjustment for Heterogeneous Hadoop Clusters. *IEEE Syst. J.* **2016**, *10*, 471–482. [CrossRef]
51. Padillo, F.; Luna, J.M.; Herrera, F.; Ventura, S. Mining association rules on big data through mapreduce genetic programming. *Integr. Comput. Aided Eng.* **2018**, *25*, 31–48. [CrossRef]
52. Selvarani, S.; Sadhasivam, G.S. Improved cost-based algorithm for task scheduling in cloud computing. In Proceedings of the 2010 IEEE International Conference on Computational Intelligence and Computing Research, Coimbatore, India, 28–29 December 2010; pp. 1–5. [CrossRef]
53. Pan, J.S.; Dao, T.K.; Pan, T.S.; Nguyen, T.T.; Chu, S.C.; Roddick, J.F. An Improvement of Flower Pollination Algorithm for Node Localization Optimization in WSN. *J. Inf. Hiding Multimed. Signal Process.* **2017**, *8*, 486–499.
54. Arul Xavier, V.; Annadurai, S. Chaotic social spider algorithm for load balance aware task scheduling in cloud computing. *Clust. Comput.* **2019**, *22*, 287–297. [CrossRef]
55. Madni, S.H.H.; Abd Latiff, M.S.; Abdullahi, M.; Abdulhamid, S.M.; Usman, M.J. Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment. *PLoS ONE* **2017**, *12*, e0176321. [CrossRef]
56. Song, P.C.; Chu, S.C.; Pan, J.S.; Yang, H. Simplified Phasmatodea population evolution algorithm for optimization. *Complex Intell. Syst.* **2021**, 1–19. [CrossRef]
57. Chu, S.C.; Tsai, P.w.; Pan, J.S. Cat Swarm Optimization. In *PRICAI 2006: Trends in Artificial Intelligence*; Yang, Q., Webb, G., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 854–858.
58. Sun, Y.; Pan, J.S.; Hu, P.; Chu, S.C. Enhanced Equilibrium Optimizer algorithm applied in job shop scheduling problem. *J. Intell. Manuf.* **2022**, 1–27. [CrossRef]
59. Pan, J.S.; Nguyen, T.L.P.; Ngo, T.G.; Dao, T.K.; Nguyen, T.T.T.; Nguyen, T.T. An Optimizing Cross-Entropy Thresholding for Image Segmentation based on Improved Cockroach Colony Optimization. *J. Inf. Hiding Multimed. Signal Process.* **2020**, *11*, 162–171.
60. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. *Comput. Intell. Lab. Zhengzhou Univ. Zhengzhou China Tech. Rep. Nanyang Technol. Univ. Singap.* **2013**, *635*, 490.
61. Chu, S.C.; Xu, X.W.; Yang, S.Y.; Pan, J.S. Parallel fish migration optimization with compact technology based on memory principle for wireless sensor networks. *Knowl. Based Syst.* **2022**, *241*, 108124. [CrossRef]
62. Pan, J.S.; Wang, X.; Chu, S.C.; Nguyen, T. A multi-group grasshopper optimisation algorithm for application in capacitated vehicle routing problem. *Data Sci. Pattern Recognit.* **2020**, *4*, 41–56.