IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 23, 2020, accepted March 3, 2020, date of publication March 19, 2020, date of current version March 31, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2981972*

# Improved PSO Algorithm Integrated With Opposition-Based Learning and Tentative Perception in Networked Data Centres

**ZHOU ZHOU[1], FANGMIN LI[1], JEMAL H. ABAWAJY[2], (Senior Member, IEEE), AND CHAOCHAO GAO[3]**

[1]Department of Mathematics and Computer Science, Changsha University, Changsha 410083, China
[2]School of Information Technology, Deakin University, Geelong, VIC 3217, Australia
[3]School of Information Science and Engineering, Hunan Normal University, Changsha 410006, China

Corresponding authors: Zhou Zhou (zhouzhou03201@126.com) and Fangmin Li (lifangmin@whut.edu.cn)

This work was supported in part by the Hunan Province Key Laboratory of Industrial Internet Technology and Security under Grant 2019TP1011, in part by the National Natural Science Foundation of China under Grant 61772088, in part by the China Postdoctoral Science Foundation under Grant 2018M642974, in part by the Natural Science Foundation of Hunan Province, China, under Grant 2019JJ50689, and in part by the Scientific Research Project of Education Department of Hunan Province under Grant 18B412.

**ABSTRACT** Particle swarm optimization (PSO) algorithms have low-quality initial particle swarm, which is generated by a random method when handling the problem of task scheduling in networked data centres. Such algorithms also fall easily into local optimum when searching for the optimal solution. To address these problems, this study proposes combining opposition-based learning (OBL) and tentative perception (TP) with PSO; the proposed method is called OBL–TP–PSO. This algorithm uses reverse learning to generate the initial population, such that the quality of the initial particle swarm can be improved. Before the particle speed and location are updated, the TP method is used to search for the individual optimum around each particle, thereby reducing the possibility of missing the potential optimal solution during the process of searching. In this manner, the problem in which the PSO algorithm easily falls into the local optimal is effectively solved. To evaluate the performance of the proposed algorithm, simulation experiments are performed on CloudSim toolkit. Experimental results show that in comparison with other algorithms (namely, Min-Min, Max-Min and PSO algorithm), the proposed OBL–TP–PSO algorithm has better performance in terms of the total execution time, load balancing and quality of service.

**INDEX TERMS** Big data processing, high-performance data processing, networked data centre, opposition-based learning, tentative perception.

## I. INTRODUCTION

Cloud computing [1], [2] is an emerging technology that has been around for a few years. This new paradigm has provided numerous benefits for the corporate world. Most of the corporate world is now migrating to the cloud computing environment from normal computing solutions. The benefits of cloud computing include achieving economies of scale by increasing volume output or productivity with few people, reducing the cost of technology infrastructure and maintaining easy access to information with minimal upfront spending. Globalising the workforce in an affordable manner, such as granting people cloud access worldwide, is possible provided an Internet connection. Streamline processes accomplish tasks in less time with few people and reduce capital costs (e.g. spending less on hardware, software or licensing fees). Cloud computing improves accessibility by accessing resources anytime and anywhere, thereby making life considerably easy; it can monitor projects effectively by staying within budget and ahead of completion cycle times; and it requires few personnel training. It takes few people to do accomplish many tasks on a cloud, with a minimal learning curve on hardware and software issues. Moreover, cloud computing minimises the licensing of new software, such as stretch and grow, without the need to purchase expensive software licenses or programs; it also improves flexibility by changing directions without serious 'people' or 'financial' issues at stake [3], [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Shagufta Henna.

Given these benefits from cloud computation environments, cloud computing still faces enormous challenges. One of the challenges is task scheduling optimisation as per different cloud consumer demands. In a cloud environment, many tasks and resources exist; thus, task scheduling aims to meet optimal makespan, load balancing and quality of service (QoS) requirements [5], [6]. Many task algorithms, such as those that are based on particle swarm optimisation (PSO) [7]–[9], ant colony optimisation (ACO) [10]–[12], greedy strategy [13]–[16], and genetic algorithm (GA) [17]–[20].

PSO [7]–[9] is a type of evolutionary algorithm that is similar to simulated annealing. It begins from random solutions through the iterative search for the optimal solution. It can also evaluate the quality of the solution through fitness functions. Different from GA, PSO includes no crossover and mutation operators. This algorithm has attracted the attention of the academic circle for its advantages of easy implementation, high precision and fast convergence; and its advantages are shown in solving practical problems. For example, Almaamari and Omara [8] proposed a dynamic adaptive PSO algorithm (called DAPSO) to improve the performance of the PSO algorithm by decreasing the makespan and increasing resource utilisation. However, PSO algorithms have low-quality initial particle swarm, which is generated by a random method when handling the problem of task scheduling in cloud computing; such algorithms also fall easily into local optimum when searching for the optimal solution [9].

The idea of ACO [10]–[12] comes from the ant foraging behaviour in life. Ants are often successful at finding the shortest distances to acquire food and they also send a unique signal along the way to attract other ants to find the food. This special behaviour of ants is applied to job scheduling and a 'signal' is used to obtain the optimal solution. ACO algorithms can realise the optimal route selection scheduling between jobs and resources through signal information. Dai *et al.* [12] proposed a novel algorithm based on the combination of ACO and GA to achieve the performance goal of a task scheduling algorithm. The experimental results showed that the proposed algorithm surpassed other algorithms in terms of QoS and load balance (LB).

Other task scheduling algorithms include simulated greedy strategy [13]–[16] and GA [17]–[20]. Etminani and Naghibzadeh [13] proposed a novel algorithm called Min-Min in handling the problem of task scheduling to improve the resource utilisation and minimise the task execution time. The weakness of the Min-Min algorithm is that it prefers scheduling small tasks. Different from the Min-Min algorithm, Moreno and Alonso-Conde [14] proposed an algorithm called Max-Min, which prefers to schedule big tasks. The experimental results showed that the proposed algorithm could achieve a good performance in terms of LB and delivering QoS for users. GAs [17]–[20] leverage related technologies, including coding, decoding, reproduction, crossover and mutation approaches, to optimise task assignment and execution, with the intention to reduce task execution time. Manasrah and Ali [20] proposed a novel algorithm called GA–PSO algorithm based on the combination of GA and PSO algorithm to improve the efficiency of task allocation and minimize the cost and makespan. However, to some extent, the aforementioned task algorithms do not fulfil all the previously mentioned objectives, such as optimal makespan, load balancing and QoS requirements. In many cases, load balancing is ignored or avoided. In some case, optimal makespan and QoS are met, although a scope of QoS requirements can still be improved.

In this study, we propose an improved PSO algorithm integrated with opposition-based learning (OBL) and tentative perception (TP). We call this algorithm OBL–TP–PSO. The proposed algorithm optimises the cloud computational task scheduling efficiently to meet all the objectives of task scheduling. The significance of the proposed algorithm is that it finds the optimal solution at minimum number of iterations compared with existing task algorithms. We conduct extensive simulation experiments using the CloudSim toolkit. The outcome of the simulation experiments shows that OBL–TP–PSO significantly performs better than other existing task scheduling algorithms. Overall, the contributions of this study can be summarised as follows.

(1) A formalised definition of LB and QoS indicator is presented. The QoS indicator considers several factors, such as the total task execution time, rate of convergence and LB.
(2) The OBL–TP–PSO algorithm is proposed on the basis of the combination of OBL and TP.
(3) The efficiency of the proposed algorithm is evaluated and verified through extensive experiments through using the CloudSim toolkit.

The remainder of this paper is organized as follows. Section 2 introduces the methods. Section 3 presents the results and discussion. Finally, Section 4 concludes this study.

## II. OBL–TP–PSO ALGORITHM

The PSO algorithm is a random search algorithm that simulates the behaviour of birds in the natural world, that is, searching for food. It is a branch of evolutionary algorithm that mainly finds the optimal solution through a group collaboration mechanism. In comparison with GA, ACO, simulated annealing and tabu search algorithms, PSO has various advantages, including fewer parameters, easier realisation and higher rate of convergence. Therefore, PSO is widely used in solving engineering optimisation problems, such as task scheduling, combinatorial optimisation and image processing.

### A. PRINCIPLE OF PSO ALGORITHM

In the traditional PSO algorithm, assuming that $m$ particles form a particle swarm in a D-dimensional solution domain, the position of particle $i$ at the $k$-th iteration cycle is expressed as

$$X_i^k = (x_{i1}^k, x_{i2}^k, \cdots, x_{id}^k) \quad x_{id}^k \in [X_{mind}, X_{maxd}] \tag{1}$$

where $X_i^k$ denotes the position of particle $i$ at the $k$-th iteration, and $[X_{mind}, X_{maxd}]$ refers to the range reachable by the moving particle.

Particle velocity is expressed as

$$V_i^k = (v_{i1}^k, v_{i2}^k, \cdots v_{id}^k) \quad v_{id}^k \in [V_{mind}, V_{maxd}] \tag{2}$$

where $V_i^k$ is the velocity of particle $i$ at the $k$-th iteration, and $[V_{mind}, V_{maxd}]$ denotes the range of velocity of the moving particle.

The optimal position of particle $i$ that has been found up to the present moment is the individual extremum and is recorded as Pbesti. Meanwhile, the optimal position of the entire particle swarm that has been found up to the present moment is the global extremum and is recorded as Gbest.

When the individual and group optimal solutions are found, the particle velocity and position are updated using Formulas (1)–(4).

$$v_{ik}^{t+1} = v_{ik}^t + c_1 r_1 (Pbest_i - x_{ik}^t) + c_2 r_2 (Gbest - x_{ik}^t) \tag{3}$$

$$x_{ik}^{t+1} = x_{ik}^t + v_{ik}^t \tag{4}$$

where $c_1$ and $c_2$ are the learning factors, which are also called acceleration constants and represent the influences of individual and group optimal solutions on the previous velocity, respectively; and $r_1$ and $r_2$ are random numbers within the range of [0, 1]. Formula (3) is a particle swarm velocity update formula, which can be divided into three parts. First is the 'inertia' part, which reflects the particle's inertia and represents the particle's trend of maintaining its previous velocity; second is the 'cognition' part; and third is the 'social' part, which reflects the collaboration among particles. Formula (4) is a particle position update formula. Particle movement in PSO algorithm is shown in Fig. 1.

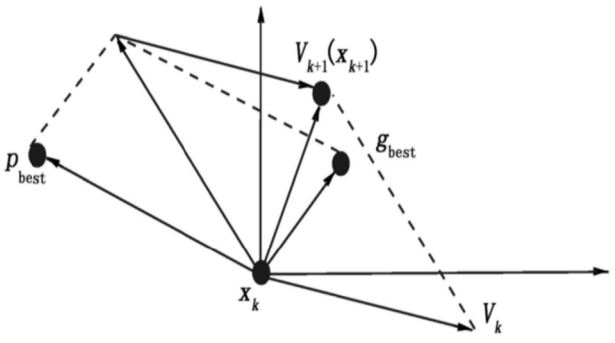



**FIGURE 1.** Particle movement.

### B. PROCEDURE OF PSO ALGORITHM

On the basis of the principle of PSO algorithm, its basic procedure is as follows.

(1) Generate initial particle swarm. Assume that the population size is $N$, and the position and velocity of each particle are $X_i$ and $V_i$, respectively.
(2) Calculate the fitness value in each particle state ($F_{fitness}[i]$).
(3) Compare the fitness value of each particle ($F_{fitness}[i]$) with the corresponding individual extremum (Pbest[i]). If the fitness value of the particle is better than the individual extremum, then replace Pbest[i] with $F_{fitness}[i]$. For example, for a minimum optimisation problem, if $F_{fitness}[i] <$ Pbest[i], then replace Pbest [i] with $F_{fitness}[i]$; for a maximum optimisation problem, if $F_{fitness}[i] >$ Pbest[i], then replace Pbest[i] with $F_{fitness}[i]$.
(4) Compare each particle's fitness value ($F_{fitness}[i]$) with the global extremum (Gbest). If the particle's fitness value is better than the global extremum, then replace Gbest with $F_{fitness}[i]$.
(5) Update particle velocity $V_i$ and position $X_i$ according to Formulas (1)–(4).
(6) If the end condition is met, then exit from the iteration cycle; otherwise, return to Step (2), where the end condition usually refers to the maximum number of iteration cycles''.

### C. OBL–TP–PSO ALGORITHM

#### 1) STRATEGY OF PARTICLE SWARM INITIALISATION BASED ON OBL

##### a: PRINCIPLE OF OBL

Assume that $x^i = m + n - x_i$, where $x_i$ denotes the original data; $x^i$ is the target data; and $m$ and $n$ refer to the lower and upper limits of range ($m$, $n$), respectively. Then, $x^i$ is the opposite of $x_i$. This principle is extended to multidimensional vectors. Assume that the value range of each dimension of $n$-dimensional vector $X_i = (x_1, x_2, x_3, \ldots, x_n)$ is $(m_k, n_k)$. Then, the opposite vector obtained from $X_1$ is $X^i = (m_1 + n_1 - x_1, m_2 + n_2 - x_2, m_3 + n_3 - x_3, \ldots, m_k + n_k - x_n)$.

At this point, assuming that $n$ particles are present in the space, the population formed by these particles is called the original population. $P_i$ denotes particle $i$ at a certain velocity and position in the space; its velocity vector is expressed as $V_i$, and its position vector is expressed as $S_i$. Then, particle $i$ is expressed as $P_i = \{V_i, S_i\}$. The OBL method can show that having an opposite particle $P^i = \{V^i, S^i\}$ in the space is inevitable. The population formed from the original population using OBL is called the opposite population.

##### b: PRINCIPLE OF PARTICLE SWARM INITIALISATION BASED ON OBL

Traditional PSO algorithms are highly dependent on the initial particle swarm when finding the solution. Therefore, using an appropriate method to optimise the initial particle swarm plays a critical role in finding the solution to the problem. When a PSO algorithm solves a task scheduling problem, the computation time of the algorithm is related to the distance between other individuals and the optimal individual in the population. The closer other individuals are to the optimal individual, the easier it will be for the algorithm to converge quickly. For PSO algorithms, the particles generated by a pure random function are not estimated; thus, predicting

the rate of convergence is impossible. The probability that a certain particle will be close to the optimal particle is equal to the probability that the opposite particle of this particle will also be close to the optimal particle. Therefore, the initial particle swarm formed by better particles, which are preferentially selected from the particle population considering the opposite particle of each particle, will also be of better quality. The detailed procedure is as follows.

1) Randomly initialise n particles to form an initial population.
2) Generate n opposite particles through OBL to form an opposite population.
3) Calculate the fitness values of the original and opposite particles in the initial and opposite populations, and select better particles to form an initial population for the PSO algorithm.

### 2) SEARCH STRATEGY BASED ON TP

When the PSO algorithm searches for the optimal solution to a problem in the solution domain, all particles will "fly" to the optimal position in the particle swarm at the time of each iteration and update, such that the algorithm can converge quickly. The update pattern is shown in Fig. 2.

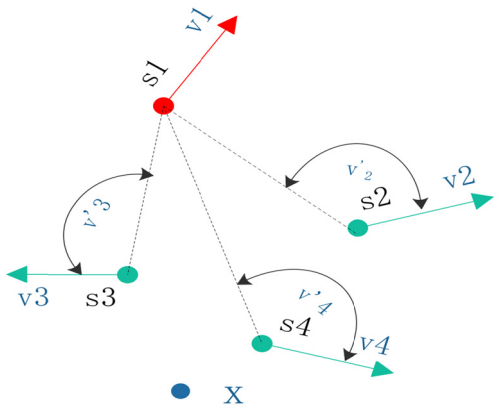



**FIGURE 2.** The update pattern of particles.

We assume that four particles form a particle swarm in the solution domain, in which the optimal solution to a certain problem can be found. The current positions and velocities of these particles are {s1, s2, s3, s4} and {v1, v2, v3, v4}, respectively. To further address the problem, we assume that the fitness value corresponding to the position of each particle is the individual extremum that has been found up to the present moment, and the fitness value corresponding to s1 is the optimal fitness value of particles in the swarm. On the basis of the particle velocity and position update formula, s2, s3 and s4 all approach towards group optimal solution s1 during update and search for the optimal solution. The arc area in Fig. 2 is the range of updated velocity. Assume the velocity update method of standard PSO algorithm is used to search for the solution in the solution domain; if point x denotes the position of an optimal solution existing around s4 under ideal conditions, which objectively exists but has not been found, then the algorithm tends to omit such objectively existing optimal solutions and become stuck in local optima. Even if x can be found at the late stage of iterative search by adjusting the values of $\omega$, $c1$ and $c_2$ in the PSO algorithm, the possibility of finding this value remains extremely low.

In response to the preceding problems, this study proposes a search method based on individual TP. Assuming that before the velocity of each particle is updated, $q$ probes at random directions are virtualised at the current position, and the size of $q$ depends on the scale of the problem and the size of the initial population. When the scale of the problem is given, the smaller the initial population is, the larger $q$ will be. The velocity and position of probe $j$ in dimension $k$ at iteration t of particle $i$ are expressed as follows:

$$V_{ik}^{j} = \omega v_{ik}^{t} + c_1 R_1 (Pbest_i - x_{ik}^{t}) + c_2 R_2 (Gbest - x_{ik}^{t}) \quad (5)$$

$$X_{ik}^{j} = x_{ik}^{t} + V_{ik}^{j} \quad (6)$$

$$R_1 = a - r_1 \quad (7)$$

$$R_2 = b - r_{21} \quad (8)$$

On the basis of Formulas (5) and (6), the velocity of the initialised probe of particle $i$ is also composed of three parts, that is, the constraints of the original particle's velocity, of the individual historical optimal solution and of the group optimal solution on the probe. Parameters $c_1$ and $c_2$ denote the acceleration factors; $r_1$ and $r_2$ are random numbers within the range of [0, 1]; and parameters $a$ and $b$ are constants within the range of [0, 1]. By setting the values of $a$ and $b$, the probability that the probe will move in the direction opposite to that of Pbest and Gbest under the original particle conditions can be controlled.

In comparison with the velocity update formula in the PSO algorithm, the initialised probe has a larger space to move in. If the fitness value of the $j$-th initialised probe of particle $i$ is better than the Pbest of this particle, then this particle is replaced by probe $j$.

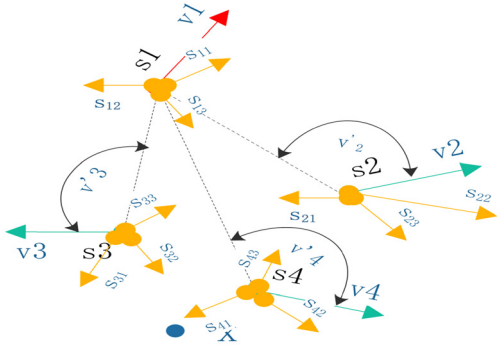



**FIGURE 3.** Tentative perception model article movement.

The model of the search strategy based on TP is shown in Fig. 3. Assuming that $q = 3$, three probes at random directions {s41, s42, s43} are distributed around particle s4 before its velocity and position are updated; that is, before

**Algorithm 1** OBL–TP–PSO Algorithm

```
Input: particle swarm size N, probe number q, task number M
Output: Gbest and i
For each particle i
        Initialize velocity Vi and position Xi for each particle i;
        Initialize velocity Vi' and position Xi' for each particle i using OBL;
        Evaluate positions Xi and Xi' of particle i with the fitness function fit (Xi);
        Choose the better position to be Xi and set Pbest_i = Xi;
End for
Gbest = Min{fit(Pbest_i)};
While i < maxTimes
        For i = 1 to N
            new q probes from particle i;
            Evaluate each probe with fitness function fit();
            Select the best one from Pbest_i and probes and set it to be particle i;
            Update the velocity and position of particle i;
            If fit (xi) < fit (Pbest_i)
              Pbest_i = Xi;
            End if
            If fit (Pbest) < Gbest
              Gbest = Pbest_i;
            End if
        End for
End while
return Gbest and i;
```

it flies to the swarm, their fitness values are calculated and the optimal probe is selected. If the fitness value corresponding to this probe is better than that corresponding to s4, then s4 is replaced by the position of the optimal probe and the particle velocity and position are then updated for the next iteration cycle. If the fitness values of all probes are worse than that of s4, then s4 will remain unchanged and the particle velocity and position are updated for the next iteration cycle. This search strategy tentatively perceives any position around each particle that is more optimal than the position of this particle to prevent such positions from being omitted during the search for optimal solution. This search method based on TP improves the capability of the PSO algorithm to find local optima while preventing the algorithm from being stuck.

#### 3) OBL–TP–PSO ALGORITHM

Algorithm 1 shows the pseudo-code of the improved PSO algorithm integrated with OBL and TP (OBL–TP–PSO) algorithm. The procedure for OBL–TP–PSO is as follows.

(1) Initialise $n$ original particles to form an initial population.
(2) Generate the opposite particles of $n$ original particles through OBL.
(3) Select better particles from the original and opposite particles to form an initial particle swarm.
(4) Calculate the fitness value of each particle using the fitness function.
(5) Generate $q$ probes around each particle to perceive any position that is better than this particle's position.
(6) Calculate the fitness value corresponding to each probe. Select the optimal fitness value and compare it with the fitness value of the present particle. If the former is better than the latter, then replace the present particle with the probe.
(7) Update the particle velocity using Formula 2.
(8) Update the particle position using Formula 1.
(9) Determine whether the cycle end condition is met. If met, then exit from the iteration cycle; otherwise, return to Step (4).

#### 4) OBL–TP–PSO ALGORITHM SOLVING TASK SCHEDULING PROBLEM IN CLOUD COMPUTING

##### *a: ENCODING PROBLEM*

Encoding plays an important role for the PSO algorithm. Currently, many encoding methods exist for intelligent optimisation problems. In this study, particles are encoded sequentially. Assume that $N$ tasks and $M$ resources exist, and $N > M$. Then, the particle encoding and decoding processes are as follows.

Assume that $N = 10$; $M = 5$; and number of tasks and resources is expressed as $(1, 2, \ldots, 10)$ and, respectively. Then, a sequential code A = (1, 3, 5, 2, 4, 5, 1, 3, 4, 3) exists, where the order of digits in the code corresponds to the task number, and the digits in the code represents the resource number. At this time, variable A can be deemed as a particle in the solution domain and corresponds to a certain task scheduling scheme. The code expression is shown in Table 1.

**TABLE 1.** Encoding.

| Task number | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| Resource number | 1 | 3 | 5 | 2 | 4 | 5 | 1 | 3 | 4 | 3 |

**TABLE 2.** Decoding.

| Task number | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Resource number | 1 7 | 4 | 2 8 10 | 5 9 | 3 6 |

The code above can be expressed as follows. For execution, Task 1 is assigned to Resource 1, Task 2 is assigned Resource 3, Task 3 is assigned to Resource 5 for execution, and so on. Encoding corresponds to decoding. We can decode the particles to understand the status of task execution on each resource. The results of task scheduling after decoding of A are shown in Table 2.

After A is decoded, the results of task scheduling can be observed clearly. The tasks assigned to Resources 1–5 are {1, 7}, {4}, {2, 8, 10}, {5, 9}, and {3, 6}, respectively.

### b: SELECTION OF FITNESS FUNCTION

PSO and improved PSO algorithms require fitness values to determine the soundness of a particle position, which is usually a multi-objective optimisation problem. However, multiple objectives are mutually contradictory and conflict with one another in most cases of multi-objective optimisation. Therefore, the best optimisation method measures each objective and obtains the corresponding extremum based on actual conditions. In this study, total task execution time is used as the main indicator of the algorithms' performance.

Task scheduling aims to complete the tasks in a most efficient way within the shortest time. The shorter the task completion time is, the larger the value of the objective function and the higher the fitness of particles in the swarm will be. Therefore, the objective function must be initially defined, and Formula (1) is substituted into fitness function $f(x)$ to determine the fitness value.

Assume that $T = \{T1, T2, \ldots, Tr\}$ denotes r inter-independent tasks; and $R = \{R1, R2, \ldots, Rs\}$ denotes $s$ computing resources in the cloud data centre, where $r > s$, the exeTime matrix of r x s denotes the time required to complete the execution of task queue on each computing resource, $T(Ti)$ denotes the size of task $Ti$ and Mips($Rj$) denotes the computing capacity of resource $Rj$. Then, the time required to complete the execution of task $i$ on resource $j$ is expressed as follows:

$$exeTime(i, j) = \frac{T(Ti)}{Mips(Rj)} \tag{9}$$

Assuming that n tasks are assigned to resource j, the time required for resource j to complete all tasks is:

$$\text{com Time (j)} = \sum_{i=1}^{n} \text{exe Time } (i, j) \tag{10}$$

where $i$ denotes task $i$ executed on resource $j$, and $n$ denotes the total number of tasks assigned to resource $j$. Then, the total time of execution of all tasks in the system is:

$$AllexeTime(i) = Max_{j=1}^{s}(comTime(j)) \tag{11}$$

Fitness represents the soundness of particle position. In this study, the network bandwidth and the tasks' demands for the resources' performance and security during task scheduling are ignored, and only the total time of task scheduling is considered. In this case, the fitness function is defined as:

$$F(i) = \frac{1}{AllexeTime} \tag{12}$$

where $F(i)$ denotes the fitness value corresponding to particle $i$. A short total task execution time, expressed by $AllexeTime$, indicates small task makespan, large corresponding fitness value and good particle position.

## III. RESULTS AND DISSCUSSION

In this experiment, Java was used as the program development language in Eclipse 4.3, and CloudSim 3.3.2 [21], [22] was used as the cloud computing simulation platform. The experimental environment was configured as follows: Intel(R) Core(TM) i5 Dual-Core 3.4 GHz CPU, 4 GB internal memory and Windows 7 operating system. The proposed task scheduling algorithm integrated with OBL and TP, that is, OBL–TP–PSO algorithm, was analysed and compared with Min-Min [13], Max-Min [14] and PSO [8] algorithms.

The parameters of the PSO algorithm were adjusted several times. The results indicated that when $w = 0.5$ and $c_1 = c_2 = 1$, the PSO algorithm could quickly find an accurate solution. The parameters of OBL–TP–PSO algorithm were set similar to those of the PSO algorithm; and parameters $q$ and $a$ in the OBL–TP–PSO algorithm were set as 3 and 0.5, respectively. Such setting represents that before each particle flies to Pbest and Gbest, three probes could be generated randomly at the original position for tentative search. In this manner, we could determine if any solution that is more optimal than the present position exists. The probability that each probe would search along the direction opposite to that of the group and individual optimal solutions was 50%.

For this experiment, the number of tasks was set to 100, 200 and 500; the size of tasks was generated as random

**TABLE 3.** Total task execution time when task number is 100.

| Algorithm | Total task execution time | | |
|---|---|---|---|
| | Minimum value | Maximum value | Average value |
| Min-Min | 36725.98 | 36725.98 | 36725.98 |
| Max-Min | 35619.99 | 35619.99 | 35619.99 |
| PSO | 35186.98 | 37596.98 | 36129.98 |
| OBL-TP-PSO | 34652.79 | 34690.32 | 34676.63 |

**TABLE 4.** Total task execution time when task number is 200.

| Algorithm | Total task execution time | | |
|---|---|---|---|
| | Minimum value | Maximum value | Average value |
| Min-Min | 70570.22 | 70570.22 | 70570.22 |
| Max-Min | 69433.97 | 69433.97 | 69433.97 |
| PSO | 72647.97 | 76226.46 | 73871.63 |
| OBL-TP-PSO | 68919.91 | 68938.57 | 68930.45 |

**TABLE 5.** Total task execution time when task number is 500.

| Algorithm | Total task execution time | | |
|---|---|---|---|
| | Minimum value | Maximum value | Average value |
| Min-Min | 169856.43 | 169856.43 | 169856.43 |
| Max-Min | 169162.88 | 169162.88 | 169162.88 |
| PSO | 175125.40 | 177330.35 | 176254.47 |
| OBL-TP-PSO | 168167.32 | 168180.39 | 168174.8 |

numbers within the range of [100, 1000000]; the number of VMs was set to 5; and the processing capacities of single-core processors were {100, 200, 300, 400, 500} [6]. Under the same conditions, each random search algorithm was operated for 10 times to determine the mean. Finally, the performance of the algorithms was analysed from the perspectives of total task execution time, rate of convergence, LB and QoS.

### A. TOTAL TASK EXECUTION TIME

The total task execution times when the number of tasks is 100, 200 and 500; the number of resources is 5; and the size of particle swarm is 100, 200 and 500 are summarised in Tables 3–5, respectively.

The experimental data from Tables 3–5 show that in comparison with PSO, Min-Min, and Max-Min algorithms, the OBL–TP–PSO algorithm approximately saves 5%, 2% and 1% total execution time, respectively. In other words, the solution found by the OBL–TP–PSO algorithm is more accurate than the other algorithms. The reasons are as follows. For the Min-Min or the Max-Min algorithm, the total task execution time is based on individual task execution time. The two algorithms pursue the minimum total completion time of each task when it is scheduled and cannot guarantee the total time of execution of the entire task group. Meanwhile, the OBL–TP–PSO algorithm can search for possible individual optimal solution around the particle by means of TP before each particle flies to the group optimal solution; moreover, a probability exists that the probe of the particle would search along the direction opposite to that of the group and individual optimal solutions. In this manner, the algorithm can be prevented from being stuck in local optima, and the search capability of each particle can be enhanced. Therefore, the OBL–TP–PSO algorithm can find a more accurate solution than the other three algorithms.

### B. RATE OF CONVERGENCE

The number of iterations for the PSO and OBL–TP–PSO algorithms under the conditions of different numbers of tasks were analysed and compared. The experimental results are as follows. The comparisons between the experimental results of the PSO and OBL–TP–PSO algorithm when the number of tasks is 100, 200 and 500 and the population size is 100, 200 and 500 are shown in Figs. 4–6, respectively.

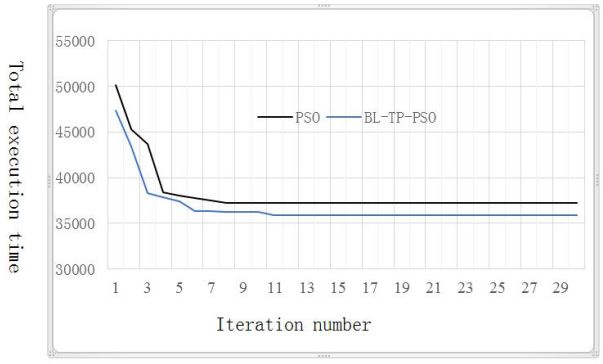



**FIGURE 4.** The iteration number for two algorithms when task number is 100.

The results of the Min-Min and the Max-Min algorithms are simple, and these algorithms do not require iteration cycles to find the solution. Thus, their convergence is not considered, and only the convergence properties of the PSO

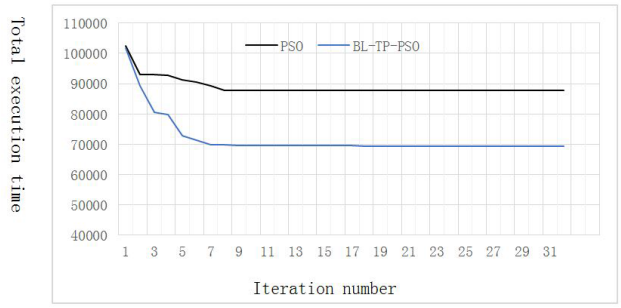



**FIGURE 5.** The iteration number for two algorithms when task number is 200.

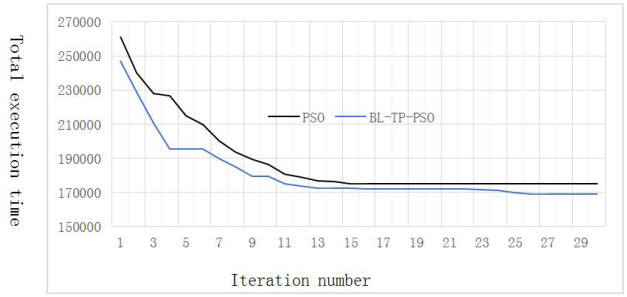



**FIGURE 6.** The iteration number for two algorithms when task number is 500.

and the OBL–TP–PSO algorithms are compared. The comparison between the preceding experimental results indicates that the OBL–TP–PSO algorithm converges more quickly than the PSO algorithm at the early stage; after the same number of iterations under the same conditions, the total task execution time of the former is shorter than that of the latter; and at the late stage, the OBL–TP–PSO algorithm achieves better convergence and finds a solution closer to the optimal solution.

Under the same conditions, the OBL–TP–PSO algorithm can generate initial particle swarm with higher quality compared with the PSO algorithm. In other words, the initial particles are more optimal and closer to the population and it is easier for them to find the Gbest. In addition, the TP strategy can be implemented before the last iteration cycle, and the probe can find better Gbest for the next iteration cycle. Therefore, for the same number of iterations, the OBL–TP–PSO algorithm can find better Gbest and converge more quickly compared with the PSO algorithm.

### C. LOAD BALANCE (LB)

Among task scheduling problems in cloud computing, LB is an important indicator for task scheduling evaluation and can well reflect the overall resource utilisation rate. Assuming that $M_i$ denotes the number of tasks assigned to resource $i$, and $M$ denotes the average number of tasks assigned to the resource, then LB is defined as follows:

$$LB = \sum_{i=1}^{j} (M - M_i)^2 \tag{13}$$

From this definition, the more balanced the task load on cloud computing resources is, the lower the LB level will be. The LB levels for different numbers of tasks are shown in Fig. 7. The LB levels of the PSO and OBL–TP–PSO algorithms are the mean values of solutions found by these algorithms for 10 times.

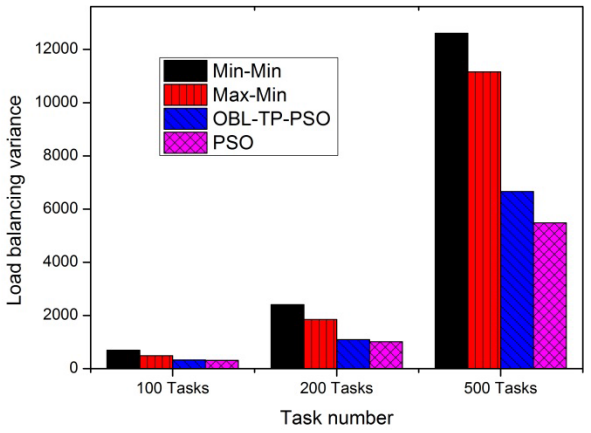



**FIGURE 7.** Load balancing variance.

As shown in Fig. 7, the Min-Min and Max-Min algorithms achieve relatively low level of LB, whereas the OBL–TP–PSO and PSO algorithms achieve relatively high level of LB.

Traditional Min-Min algorithms always preferentially assign small tasks to resources with high processing capacity for execution, thereby resulting in unbalanced task load on resources. Meanwhile, Max-Min algorithms preferentially assign large tasks to resources with high processing capacity for execution, which effectively overcomes the load unbalance problem of the Min-Min algorithm. However, these two algorithms assign tasks to system resources merely depending on the size of tasks and do not consider task combination and optimisation. Therefore, the performance of both algorithms is not ideal in terms of LB. In comparison, PSO algorithms can determine more optimal makespan and achieve better LB when solving the task scheduling optimisation problem.

## IV. CONCLUSION

To overcome the disadvantages of PSO algorithms, such as its dependence upon initial particle swarm and tendency of becoming stuck in local optima, we proposed an improved PSO algorithm called OBL–TP–PSO algorithm integrated with OBL and TP. The experimental results show that the proposed algorithm OBL–TP–PS algorithm is more accurate than the other algorithms, namely, Min-Min, Max-Min and PSO algorithms, in terms of total execution time, rate of convergence and LB.

In the future, we will conduct an actual experiment under cloud platform to evaluate the effectiveness of the OBL–TP–PSO algorithm.

## APPENDIX
## ABBREVIATION

| | |
|---|---|
| PSO: | Particle swarm optimization |
| OBL-TP-PSO: | An improved PSO algorithm based on OBL and TP algorithm |
| OBL: | Opposition-based learning |
| TP: | Tentative perception |
| QoS: | Quality of service |
| ACO: | Ant colony optimization |
| GA: | Genetic algorithm |

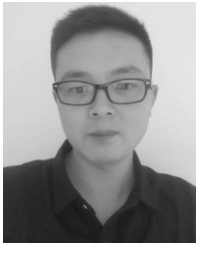
## REFERENCES

[1] B. Ahmed, A. W. Malik, T. Hafeez, and N. Ahmed, "Services and simulation frameworks for vehicular cloud computing: A contemporary survey," *EURASIP J. Wireless Commun. Netw.*, vol. 2019, no. 1, p. 4, Dec. 2019.
[2] C.-H. Zeng and K.-C. Chen, "Social network analysis facilitates cognition in large wireless networks: Clustering coefficient aided geographical routing," *IEEE Trans. Cognit. Commun. Netw.*, vol. 4, no. 3, pp. 618–634, Sep. 2018.
[3] S.-M. Zhang and A. K. Sangaiah, "Reliable design for virtual network requests with location constraints in edge-of-things computing," *EURASIP J. Wireless Commun. Netw.*, vol. 2018, no. 1, p. 65, Dec. 2018.
[4] Z. Zhou, J. H. Abawajy, F. Li, Z. Hu, M. U. Chowdhury, A. Alelaiwi, and K. Li, "Fine-grained energy consumption model of servers based on task characteristics in cloud data center," *IEEE Access*, vol. 6, pp. 27080–27090, 2018.
[5] K.-C. Wu, W.-Y. Liu, and S.-Y. Wu, "Dynamic deployment and cost-sensitive provisioning for elastic mobile cloud services," *IEEE Trans. Mobile Comput.*, vol. 17, no. 6, pp. 1326–1338, Jun. 2018.
[6] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi, and F. Li, "Minimizing SLA violation and power consumption in cloud data centers using adaptive energy-aware algorithms," *Future Gener. Comput. Syst.*, vol. 86, pp. 836–850, Sep. 2018.
[7] M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of PSO-based scheduling algorithms in cloud computing," *J. Netw. Syst. Manage.*, vol. 25, no. 1, pp. 122–158, Jan. 2017.
[8] A. Al-maamari and F. A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 5, pp. 245–256, Oct. 2015.
[9] Z. Yang, X. Qin, W. Li, and Y. Yang, "Optimized task scheduling and resource allocation in cloud computing using PSO based fitness function," *Inf. Technol. J.*, vol. 12, no. 23, pp. 7090–7095, Dec. 2013.
[10] F. E. B. Otero, A. A. Freitas, and C. G. Johnson, "Inducing decision trees with an ant colony optimization algorithm," *Appl. Soft Comput.*, vol. 12, no. 11, pp. 3615–3626, Nov. 2012.
[11] X.-J. Liu, H. Yi, and Z.-H. Ni, "Application of ant colony optimization algorithm in process planning optimization," *J. Intell. Manuf.*, vol. 24, no. 1, pp. 1–13, Feb. 2013.
[12] Y. Dai, Y. Lou, and X. Lu, "A task scheduling algorithm based on genetic algorithm and ant colony optimization algorithm with multi-QoS constraints in cloud computing," in *Proc. 7th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Aug. 2015, pp. 428–431.
[13] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorihtm for grid task scheduling," in *Proc. 3rd IEEE/IFIP Int. Conf. Central Asia Internet*, Sep. 2007, pp. 1–7.
[14] R. Moreno and A. B. Alonso-Conde, "Job scheduling and resource management techniques in economic grid environments," in *Grid Computing*. Santiago de Compostela, Spain: Springer, 2004, pp. 25–32.
[15] Y. Xu, Y. Li, Y. Zhao, H. Zou, and A. V. Vasilakos, "Selective sensing and transmission for multi-channel cognitive radio networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2011, no. 1, p. 7, Dec. 2011.
[16] Y. Mao, X. Chen, and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing," in *Proc. Int. Conf. Comput. Sci. Inf. Technol.*, 2014, pp. 457–465.
[17] C. Pizzuti, "A multiobjective genetic algorithm to find communities in complex networks," *IEEE Trans. Evol. Comput.*, vol. 16, no. 3, pp. 418–430, Jun. 2012.
[18] F. Xiao, Z. Zhang, and J. Abawajy, "Workflow scheduling in distributed systems under fuzzy environment," *J. Intell. Fuzzy Syst.*, vol. 37, no. 4, pp. 5323–5333, 2019.
[19] T. Jena and J. R. Mohanty, "GA-based customer-conscious resource allocation and task scheduling in multi-cloud computing," *Arabian J. Sci. Eng.*, vol. 43, no. 8, pp. 4115–4130, Aug. 2018.
[20] A. M. Manasrah, H. B. Ali, "Workflow scheduling using hybrid GA-PSO algorithm in cloud computing," *Wireless Commun. Mobile Comput.*, vol. 2018, no. 3, Jan. 2018, Art. no. 1934784.
[21] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose, and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Softw., Pract. Exper.*, vol. 41, no. 1, pp. 23–50, Jan. 2011.
[22] M. R. Chowdhury, M. R. Mahmud, and R. M. Rahman, "Implementation and performance analysis of various VM placement strategies in CloudSim," *J. Cloud Comput.*, vol. 4, no. 1, pp. 1–21, Dec. 2015.

**ZHOU ZHOU** received the Ph.D. degree in computer science from Central South University, Changsha, China, in 2017. He is currently a Lecturer with Changsha University. His research interests include cloud computing, high-performance computing, heuristic algorithm, energy consumption model, and virtual resource management and optimization.

**FANGMIN LI** received the Ph.D. degree in computer science from Zhejiang University, Hangzhou, China, in 2001. He is currently a Professor with Changsha University. He has authored several books on embedded systems and over 30 academic articles in cloud computing and wireless networks. He also holds ten Chinese patents. His current research interests include cloud computing, energy-efficient resource management, and wireless communications and networks.

**JEMAL H. ABAWAJY** (Senior Member, IEEE) is currently a Full Professor with the Faculty of Science, Engineering and Built Environment, Deakin University, Australia. He has delivered more than 50 keynote and seminars worldwide and has been involved in the organization of more than 300 international conferences in various capacities, including the Chair and the General Co-Chair. He is the author/coauthor of more than 250 refereed articles and supervised numerous Ph.D. students to completion. He has served on the editorial-board of numerous international journals, including the IEEE TRANSACTIONS ON CLOUD COMPUTING.



**CHAOCHAO GAO** received the M.S. degree from Hunan Normal University, in 2019. He is currently a Lecturer with Hunan Normal University. His research interests include cloud computing, high-performance computing, green computing, heuristic algorithm, energy consumption model, and virtual resource management and optimization.