Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4328185, 11 pages
https://doi.org/10.1155/2022/4328185

Research Article

Particle Swarm Optimization Algorithm Based on Information Sharing in Industry 4.0

Xiaoyang Rao and Xuesong Yan

School of Computer Science, China University of Geosciences, Wuhan 430074, China

Correspondence should be addressed to Xuesong Yan; yanxs@cug.edu.cn

Received 1 January 2022; Revised 12 January 2022; Accepted 20 January 2022; Published 10 March 2022

Academic Editor: Kalidoss Rajakani

Intelligent manufacturing is an important part of Industry 4.0; artificial intelligence technology is a necessary means to realize intelligent manufacturing. This requires the exploration of pattern recognition, computer vision, intelligent optimization, and other related technologies. Particle swarm optimization (PSO) algorithm is an optimization algorithm inspired by the foraging behavior of birds. PSO was an intelligent technology and an efficient optimization algorithm verified by a lot of research and experiments. In this paper, the traditional PSO algorithm is compared with genetic algorithms (GA) to illustrate the performance of the traditional PSO algorithm. By analyzing the advantages and disadvantages of the traditional PSO algorithm, the traditional PSO algorithm is improved through introducing into the sharing information mechanism and the competition strategy, called information sharing based PSO (IPSO). The novel algorithm IPSO was the rapid convergence speed similar to the traditional PSO and enhanced the global search capability. Our experimental results show that IPSO has better performance than the traditional PSO and the GA algorithm on benchmark functions, especially for difficult functions.

1. Introduction

Particle swarm optimization (PSO) algorithm was a bio-inspired intelligent technology proposed by Kennedy in 1995 [1]. It was a metaheuristic optimization inspired by the foraging behavior of birds [2]. Compared with the genetic algorithms (GA), we find that both algorithms are population based. But the PSO algorithm has no genetic operation, which only works on an individual chromosome. Instead, the PSO algorithm's work mechanism was the individual's exchange information with each other, so the population was optimized. Starting from a group of random solutions, the optimal solutions are obtained by searching repeatedly. When the PSO algorithm was first published, it attracted extensive attention from the optimization field scholars. Before long, it became the focus of research in the optimization field. Many scientific achievements have been made in this field [3–28]. Kulkarni and Venayagamoorthy used PSO to address wireless-sensor networks (WSN) issues such as optimal deployment, node localization, clustering, and data aggregation. Park et al. propose an improved PSO framework employing chaotic sequences combined with the conventional linearly decreasing inertia weights and adopting a crossover operation scheme to increase both exploration and exploitation capability of the PSO. Wu et al. propose a swarm-intelligence-based method-Particle Swarm Optimization (PSO) algorithm to handle the elastic parameter inversion problem. Gong et al. proposed a discrete framework of the particle swarm optimization algorithm. Based on the proposed discrete framework, a multiobjective discrete particle swarm optimization algorithm is proposed to solve the network clustering problem. Chen et al. used particle swarm optimization to determine optimal power management. Delgarm et al. proposed a mono- and multiobjective particle swarm optimization (MOPSO) algorithm to couple with Energy-Plus building energy simulation software to find a set of nondominated solutions to enhance the building energy performance. Wu et al. introduced particle swarm optimization algorithm into KNN multilabel classification and made adjustments to Euclidean distance

formula in traditional KNN classification algorithm and added weight value to each feature. Mousavi et al. proposed a modified particle swarm optimization for solving the integrated location and inventory control problems in a two-echelon supply chain network. Chatterjee et al. proposed a particle swarm optimization-based approach to train the NN (NN-PSO). The PSO is employed to find a weight vector with minimum root-mean-square error (RMSE) for the NN. The proposed (NN-PSO) classifier is capable of tackling the problem of predicting structural failure of multistoried reinforced concrete buildings via detecting the failure possibility of the multistoried RC building structure in the future.

Through research and experiments, the PSO algorithm was an efficient metaheuristic optimization algorithm [29]. PSO algorithm does not need to model the optimization problem and has strong global searchability. But, like all metaheuristics algorithms, PSO algorithm cannot guarantee finding the optimal solution. PSO algorithm is special in that it does not need to use the optimization problem's gradient. In other words, the PSO algorithm is unlike the classic deterministic optimization methods, for example, gradient descent method and the quasi-Newton method. Therefore, the PSO algorithm is widely used in partially irregularity optimization, high noise optimization, and time-varying optimization, etc. However, the disadvantage of the traditional PSO algorithm is that it falls into local optimum easily. Thus, it is needed to improve PSO to avoid this shortcoming. This paper proposes an information sharing based PSO algorithm (called IPSO, details described in Section 3). The implementation is as simple as the traditional PSO, but the experimental results for benchmark functions show that IPSO is a more efficient searching ability for global optima. The main contributions of our work are as follows:

(1) Study the performance of traditional PSO by comparing it with genetic algorithms.

(2) Analyze the shortcoming of traditional PSO and develop an improved version of PSO, called IPSO. It is verified that IPSO has better performance than traditional PSO on benchmark functions.

The rest of this paper is organized as follows. Section 2 formally introduces the traditional PSO algorithm and investigates its performance on 10 benchmark functions. In Section 3, we first analyze the reasons why traditional PSO could not always achieve optimal solutions. Based on the analysis, we develop a novel PSO (IPSO) and conduct experiments. Finally, Section 4 concludes the work of this paper.

2. Materials and Methods

2.1. PSO Algorithm. The PSO algorithm was first proposed by Eberhart and Kennedy in 1995. PSO algorithm was inspired by the flight behavior of groups of birds in search of food. PSO algorithm was an efficient metaheuristic optimization algorithm and has wide applications in the optimization field.

The traditional PSO algorithm's work mechanism is to generate a group (called swarm) of candidate solutions (called particle). The movement of these particles follows certain rules and can be expressed by equations. The best location of particles and the best location of the whole swarm are important factors. When better locations are found, the entire swarm will move to that location. Repeat this process until you find a satisfactory solution (the best location) or reach the termination condition.

Suppose that $f\colon R^n \longrightarrow R$ is the minimized function. In this function, the candidate solution is a vector of real numbers, and each candidate solution has a real number value which calculates using an objective function. The function's gradient is unknown. The purpose is to get a vector A to satisfy equation $f(A) \leq f(B)$; in here, all B are in the search space. That is to say, the global minimum that the algorithm is looking for is the solution A. The global maximum can be given by the function $h = -f$.

In the traditional PSO algorithm, each solution in the search space can be thought of as a bird called "particle." The fitness value of each particle can be calculated by the objective function. The state of each particle is determined by its velocity and location. The velocity of a particle determines its direction of flight. All the particles in the swarm search for the optimal solution through mutual cooperation and information exchange. The search process of the PSO algorithm is driven by an update of each particle state. At the initialize process, the algorithm randomly generates a group of particles (random solution), and each particle includes the information of velocity and location. The velocity and location of each particle are updated using (1) and (2). (1) has determined the best experienced location for the current particle (called P_{idb}) and the current entire swarm's best location (called P_{gdb} in (1)). Update on the velocity and location of particles repeats until the iteration's maximum number is reached. The maximum number is set at the beginning of the algorithm. Through this updating mechanism iteratively, the algorithm can find (or approach) the best solution in the solution space.

Because each particle's information is updated not only by its best experienced location (P_{idb}), but also by the whole swarm's best location (P_{gdb}), then, the particles in the swarm must exchange the information with each other. For the traditional PSO algorithm, a particle's velocity and location update equations are shown in the following equations:

$$V_{\mathrm{id}}' = \omega V_{id} + \eta_1 \mathrm{rand}()\left(P_{idb} - X_{\mathrm{id}}\right) + \eta_2 \mathrm{rand}()\left(P_{gdb} - X_{\mathrm{id}}\right), \quad (1)$$

$$X_{\mathrm{id}}' = X_{\mathrm{id}} + V_{\mathrm{id}}'. \quad (2)$$

Here, ω is the weight value of the inertia factor. It is a scaling factor related to the particle's previous velocity and the value is $0 < \omega < 1$. η_1 and η_2 are the accelerating factors; they are constant such as $\eta_1 = \eta_2 = 2$. rand() is the random function to generate the random numbers. X_{id} denotes the location of the particle id. V_{id} denotes the velocity of particle id. P_{idb} denotes the best location in the current moment the particle id was

found. P_{gdb} denotes the whole swarm's best location in the current moment found.

In (1), it is combined with three parts. The first part is the influence of the particle's previous velocity. It makes the particles have the ability to expand, that makes the global search ability of the algorithm can be enhanced. The second part represents the influence of the particle's experience knowledge called the cognition part. The third part represents the learning strategy of particles called the social part. It is the experience of learning and social cooperation process of particles with each other.

The traditional PSO algorithm's flow is described like this. First, the algorithm randomly generates a group of particles, each particle with a random initial velocity V_{id} and random location X_{id} . Then, evaluate each particle through calculating the fitness value f. Find the best location P_{idb} of X_{id} and the global best location P_{gdb} which has the best fitness value in the current entire swarm. According to (1) and (2), each particle can update its location and velocity. Then, each particle recalculates the fitness value using the objective function under its current velocity and location. The fitness value of each particle was obtained according to the recalculation, and the algorithm updates each particle's best location P_{idb} and the swarm's best location P_{gdb}. Then, use (1) and (2) again to update each particle's velocity and location. Update on the velocity and location of particles repeats until the iteration's maximum number is reached; the maximum number is set at the beginning of the algorithm.

2.2. Experiment Comparison. The traditional PSO algorithm has a high convergence rate. Clerc has presented the proof in his paper [2]. We select 10 benchmark functions for the experiment to verify the high convergence rate of the PSO algorithm and compare it with the GA. The 10 benchmark functions have a multimodal function. It is a function that is difficult to solve. And the reason why we chose it is because we want to study not only the algorithm's convergence efficiency but also the algorithm's ability to find the global optimal solutions. The 10 benchmark functions are briefly described below.

F1: Schaffer function (the function's graph shown in Figure 1).

$$\min f(x_i) = 0.5 - \frac{\left(\sin^2\sqrt{x1^2 + x2^2} - 0.5\right)}{\left[1 + 0.001\left(x1^2 + x2^2\right)\right]^2}, \quad -100 \le xi \le 100, \quad f_{\min} = 1.0. \tag{3}$$

F2: Shubert function (the function's graph shown in Figure 2).

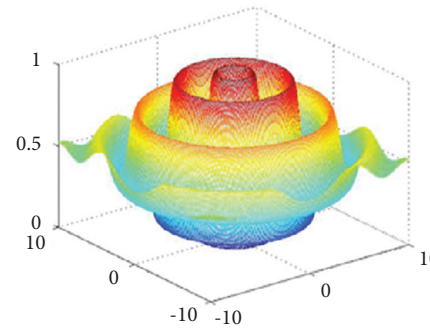

Figure 1: Schaffer function.

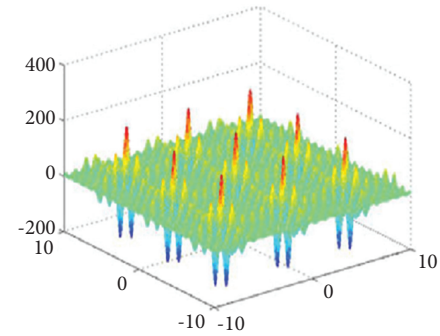

Figure 2: Shubert function.

$$\min f(x, y) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x + i]\right\} \times \left\{\sum_{i=1}^{5} i\cos[(i+1)y + i]\right\}, \quad x, y \in [-10, 10], \quad f_{\min} = -186.7309. \tag{4}$$

F3: Hansen function (the function's graph shown in Figure 3).

$$\min f(x, y) = \sum_{i=1}^{5} i\cos((i-1)x + i)\sum_{j=1}^{5} j\cos((j+1)y + j), \quad x, y \in [-10, 10], \quad f_{\min} = -176.541793. \tag{5}$$

F4: Camel function (the function's graph shown in Figure 4).

$$\min f(x, y) = \left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + \left(-4 + 4y^2\right)y^2, \quad x, y \in [-100, 100], \quad f_{\min} = -1.031628. \tag{6}$$

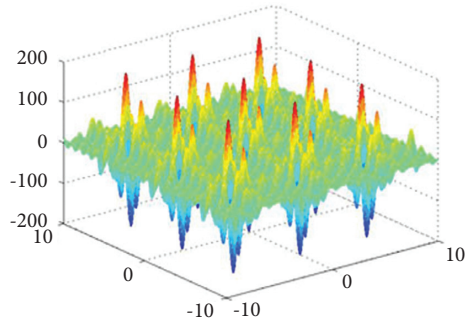

Figure 3: Hansen function.

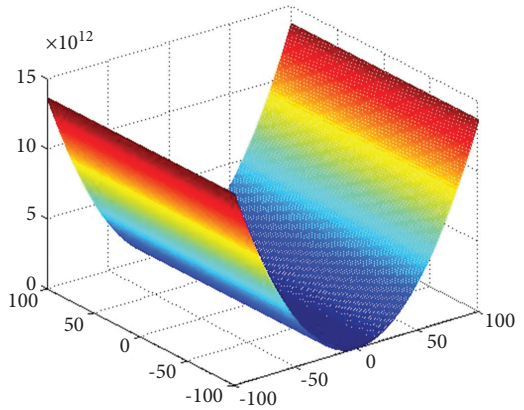

Figure 4: Camel function.

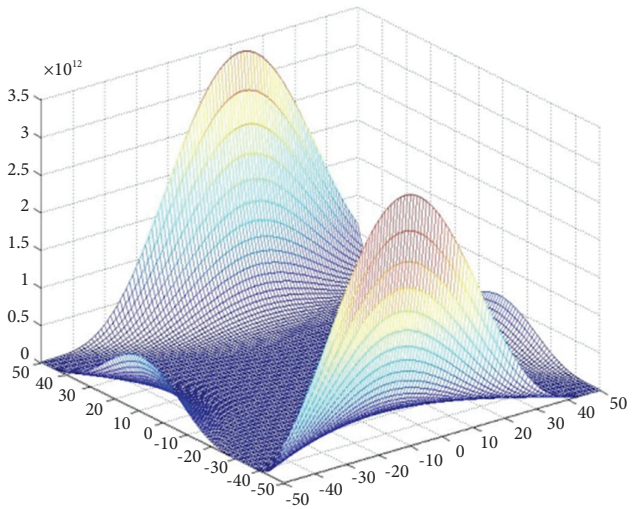

Figure 5: Function 5.

Function 5 (the function's graph shown in Figure 5).

$$\min f(x_1, x_2) = \left\{ \begin{array}{c} 1 + (x_1 + x_2 + 1)^2 \\ \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right) \end{array} \right\} \times \left\{ \begin{array}{c} 30 + (2x_1 - 3x_2)^2 \\ \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right) \end{array} \right\}, \quad x_1, x_2 \in [-50, 50], \tag{7}$$
$$f_{\min} = 3.0.$$

Function 6 (the function's graph shown in Figure 6).

$$\min f(x, y) = -\left[\begin{array}{c} x \sin(9\pi y) + \\ y \cos(25\pi x) + 20 \end{array} \right], \quad x, y \in [-10, 10],$$
$$f_{\min} = -39.944506953367. \tag{8}$$

Function 7 (the function's graph shown in Figure 7).

$$\min f(x_1, x_2) = 20 + x_1^2 + x_2^2 - 10 * (\cos 2\pi x_1 + \cos 2\pi x_2),$$
$$f_{\min} = 0. \tag{9}$$

Function 8 (the function's graph shown in Figure 8).

$$\min f(x_1, x_2) = 100 * \left(x_1^2 - x_2\right)^2 + (1 - x_1)^2, \quad x \in [-2.048, 2.048],$$
$$f_{\min} = 0. \tag{10}$$

Function 9 (the function's graph shown in Figure 9).

$$\min f(x_1, x_2) = x_1^2 + x_2^2 x_1, x_2 \in [-100, 100],$$
$$f_{\min} = 0. \tag{11}$$

Function 10 (the function's graph shown in Figure 10).

$$\min f(x_1, x_2) = 0.5x_1^2 + 0.5(1 - \cos 2x_2) + x_2^2,$$
$$f_{\min} = 0. \tag{12}$$

In order to verify the analysis of the high convergence rate of the PSO algorithm, we run 100 times algorithm experiments for the above 10 benchmark test functions. The comparison experimental results are shown in Table 1. The comparison includes the optimal solution found and the times of finding the optimal solution for each function. For example, for the difficult Function 6, GA cannot find an optimal solution for this function. The best solution GA can find is −14.786954. But PSO algorithm can find the optimal solution (−39.944506) five times in the 100 times algorithm run.

From experimental results in Table 1, we can observe that the convergence rate of the PSO algorithm is significantly higher than GA, except for the simplest function F9. For Function 9, the two algorithms can find the optimal solution 100% in 100 times algorithm run. The following conclusions can be drawn from the experimental results in Table 1: PSO algorithm has better global search ability than GA. The experimental results show that the PSO algorithm's convergence rate is better than GA. Thus, compared with GA, we prefer to apply PSO to optimization fields, such as traveling salesman problems.

From Table 1, we also notice that the PSO algorithm cannot always find the optimal solution, except the simplest function F9, although it performs better than the GA algorithm. For the functions F4, F5, and F6, in terms of the success rates of achieving the optimal solutions, PSO performs significantly better than GA. However, its success rate is also not very high. We need to analyze the reason and develop the traditional PSO in the next section.

2.3. Why PSO Could Not Always Find Optima. The above experimental results show that the reason why the PSO algorithm can converge quickly is that the cognition part and the social part in (1) can guide particles to search only around P_{gdb} and P_{idb}. By analyzing the velocity update and the location update equations of particles in the algorithm, it can be known that when the best particle in the swarm falls into the local optimum, the algorithm's information sharing mechanism will guide all the particles to this local optimum location. This will cause the entire swarm to only converge to this solution. As the entire swarm falls into the local optimum, we can find that the value of the cognition part and the value of the social part in (1) will eventually become zero. At the same time, with the execution of the iterative process, the particles' velocity will eventually drop to zero because the inertia weight is between 0 and 1 ($0 < \omega < 1$).

As all the particles in the swarm have zero velocity, the swarm has converged to one location. At this time, if the current value of P_{gdb} in the swarm is not the global optimum, this will lead to the entire swarm falling into the current optimal solution. This results in the swarm not being able to jump out of the local optimum. Especially, this occurs frequently for multimodal functions. That is the reason why the traditional PSO can only achieve the optimal solution five times out of 100 runs on function F6, although it is better than the GA algorithm, which achieves zero time out of 100 runs. To further verify this, we select three more multimodal functions (shown in Table 2 and Figures 11–13). In Table 2, *S*

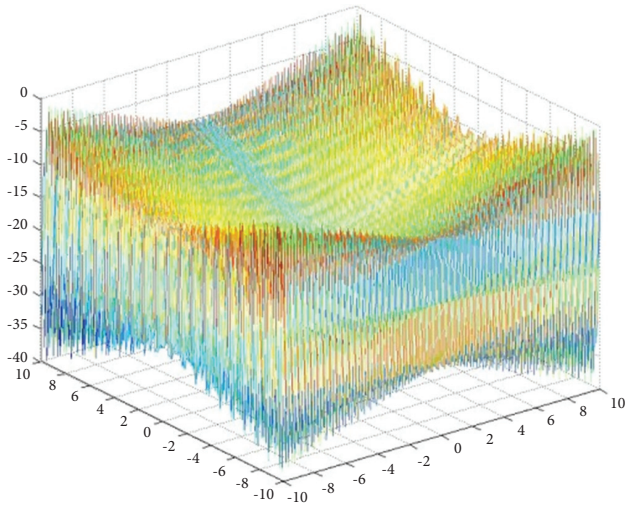

Figure 6: Function 6.

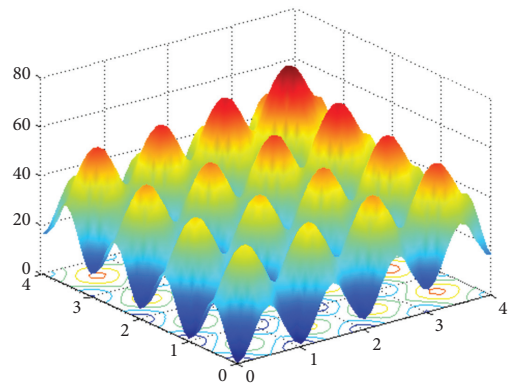

Figure 7: Function 7.

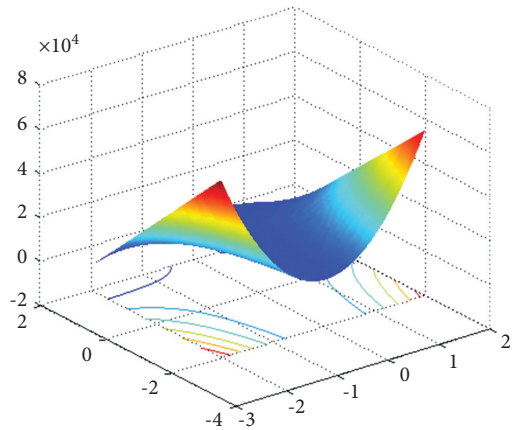

Figure 8: Function 8.

presents the range of variables; and $f_{\min}$ shows the minimum value of each function. Like what we did in Section 2, we run the traditional PSO 100 times for each function. In our experiments, the traditional PSO could not achieve the optimal solution once out of 100 runs for each function. Thus, we count the best, mean, and worst values achieved by

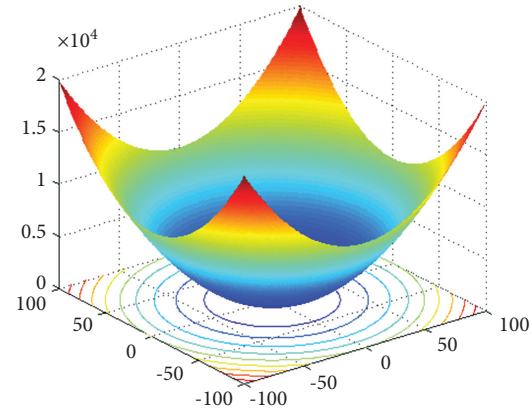

Figure 9: Function 9.

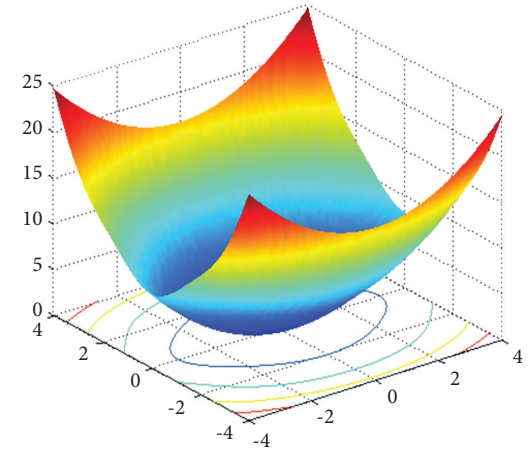

Figure 10: Function 10.

Table 1: The convergence rate of GA and PSO for the 10 benchmark functions.

Function	Algorithm	Convergence times	Optimal solution
F1	GA	72	1.0000000
	PSO	75	1.0000000
F2	GA	75	−186.730909
	PSO	80	−186.730909
F3	GA	85	−176.541793
	PSO	90	−176.541793
F4	GA	23	−1.031628
	PSO	56	−1.031628
F5	GA	16	3.000000
	PSO	25	3.000000
F6	GA	0	−14.786954
	PSO	5	−39.944506
F7	GA	90	0.000000
	PSO	95	0.000000
F8	GA	90	0.000000
	PSO	95	0.000000
F9	GA	100	0.000000
	PSO	100	0.000000
F10	GA	93	0.000000
	PSO	98	0.000000

Table 2: Three multimodal functions.

	Function	S	$f_{\min}$
F11	$f(x) = \sum_{i=1}^{n} x_i^2$	$(-100, 100)$	0
F12	$f(x) = 1/4000 \sum_{i=1}^{n} (x_i - 100)^2 - \prod_{i=1}^{n} \cos(x_i - 100/\sqrt{i}) + 1$	$(-300, 300)$	0
F13	$f(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	$(-500, 500)$	-12569.5

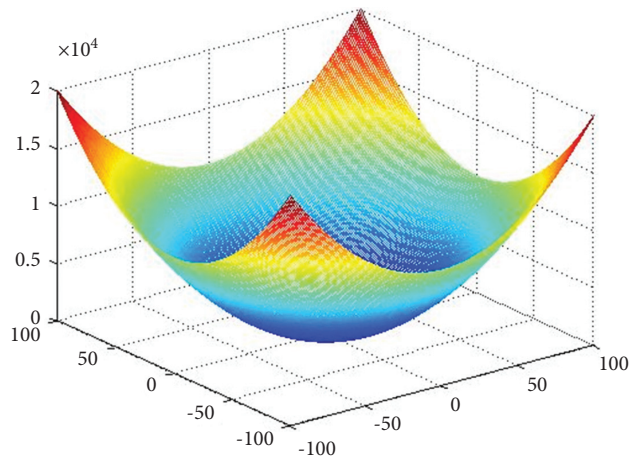

Figure 11: Function 11.

the traditional PSO in Table 3. These experimental results confirm our conjuncture: the traditional PSO algorithm falls into local optimum easily. (The GA algorithm has the same problem.)

The main reason of the traditional PSO algorithm falls into local optimum easily is because it has no mechanisms to force the particles have the ability to escape from the local optimum. This is the fatal disadvantage of the traditional PSO algorithm. This disadvantage is the reason why traditional PSO could not always find the optimal solution. With this guidance, we will develop a new solution in the next subsection, which forces the traditional PSO algorithm to avoid falling into the local optimum.

2.4. An Improved Version of PSO Algorithm. In this section, we propose an improved version of the PSO algorithm called IPSO, which aims to overcome the disadvantages of easily falling into the local optimum. The main purpose of the IPSO algorithm is must have a very strong global search ability. To achieve this purpose, a new information sharing mechanism introduces into the IPSO algorithm. This reverse mechanism lets the particles move in the opposite direction of the worst particle's location and the worst swarm's location. In this way, the global search space can be expanded and the probability of particles falling into the local optimum can be reduced.

Traditional PSO algorithm, each particle's flying direction follows a certain rule: it always approaches the location of the best particle in the swarm. From the above analysis, we can conclude that the risk of doing this is that if the best particle in the swarm cannot find the optimal solution, then it will cause the entire swarm cannot find the optimal solution. Aim at avoiding falling into local optimal solution.

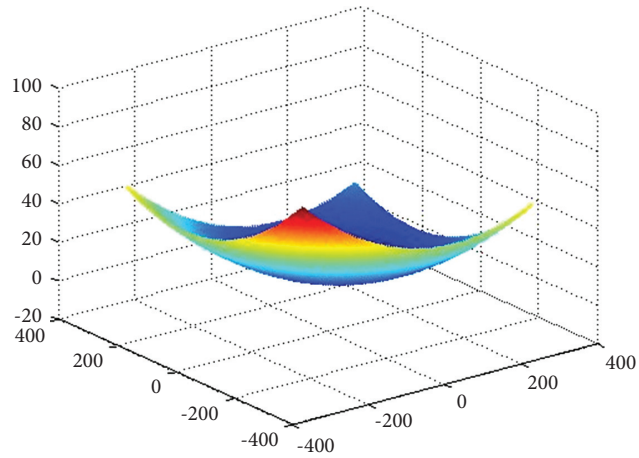

Figure 12: Function 12.

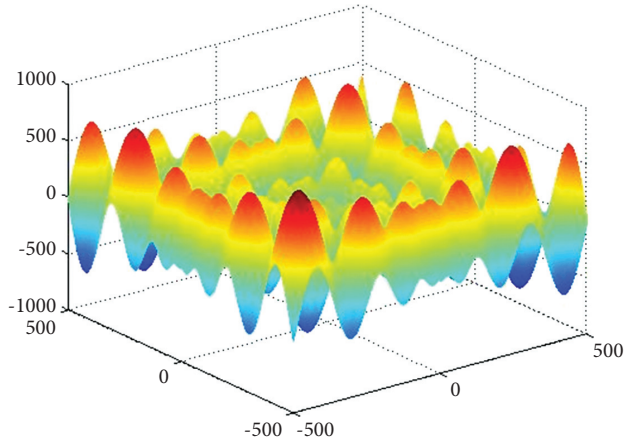

Figure 13: Function 13.

Table 3: The experimental results of PSO on the three multimodal functions.

Function	Best value	Mean value	Worst value	$f_{\min}$
F11	1495.71	4224.77	7032.89	0
F12	72.51	101.41	123.95	0
F13	−5038.62	−4005.02	−3233.13	−12569.5

The new algorithm gives each particle two flight directions (an active direction and a passive direction). This is different from the traditional PSO algorithm in which particles only have one direction of flight. The passive direction is used to store particles that fall into the local optimum. Since particles in the passive direction are not always worse than those in the active direction, the new algorithm will select the best particle from active and passive directions for the update operation. Then a competitive strategy is introduced into the new algorithm. This competitive strategy prompts the worst information to compete with the best. This makes the algorithm not only reduce the probability of falling into the local optimum solution, but also can increase the probability of converging to the optimal solution, and the time complexity is only double that of the traditional PSO algorithm.

How will the new algorithm give each particle the two flight directions (the active and the passive)? In the traditional PSO algorithm, each particle has been planned an active flight direction. This direction is still retained in the new algorithm as the active flight direction. The process that produces the passive flight direction is the same as the process that produces the active flight direction. It is well known that when a particle is in a search for the optimal solution in the solution space, it does not know where the optimum solution is. In the traditional PSO algorithm, the best location of a particle and the best location of the entire swarm will be recorded. Based on these two best locations, generate the particle's active direction through (1) and (2). We can amend the process of generating the active flight direction of particles in the traditional PSO algorithm to generate the passive flight direction of particles. In this process, we record the location of the worst particle and the worst location of the entire swarm. Based on these two worst locations, the new algorithm generates the passive direction of particle flight through (13) and (14). Equations (13) and (14) are the new velocity and location update equations of particles in the new algorithm.

$$V_{id}' = \omega V_{id} + \eta_1 \text{rand}()\left(X_{id} - P_{idw}\right) + \eta_2 \text{rand}()\left(X_{id} - P_{gdw}\right), \tag{13}$$

$$X_{\text{id}}' = X_{\text{id}} + V_{\text{id}}', \tag{14}$$

where P_{idw} represents the worst location that the particle id has recorded. P_{gdw} represents the worst location that the entire swarm has recorded.

The particle updating strategy of the IPSO is to select the best one in two flight directions (the active and the passive) for particles updating. In this way, the search space of particles can be expanded, and the premature local optimum can be avoided. Therefore, the probability of finding the global optimal solution can be improved. Since IPSO needs to calculate the velocity and location of each particle twice, the time complexity of IPSO is twice that of the traditional PSO. The framework of IPSO is explained in Algorithm 1.

3. Results and Discussion

In order to verify the effect of the IPSO algorithm, we use the same 10 benchmark functions described before to do the experiment and compare it with the traditional PSO algorithm. Again, in the experiments, the two algorithms run 100 times for every function, and the parameters are set as $\omega = 0.6$, $\eta 1 = \eta 2 = 2$. Table 4 is an experimental results comparison.

From Table 4, we can see that our IPSO achieves amazing results. It always achieves optimal solutions on six out of the 10 functions. The function F3 almost always achieves the optimal solution (97 times out of 100 runs). On the rest three difficult functions, F4, F5, and F6, our IPSO also significantly improves the success rate of finding the optimal solution. For example, for the most difficult function F6 (again, it is a multimodal function), it achieves the optimal solution 18 times out of 100 runs. The traditional PSO algorithm can only achieve five times out of 100 runs.

For the last three multimodal functions (shown in Table 2), Table 5 shows the experimental results (including the best value, the mean value, and the worst value). From Table 5, we can conclude that our IPSO also significantly improves the performance of the traditional PSO algorithm. For these three difficult functions, our IPSO can achieve very close to the optimal solutions shown in the last column (only a tiny difference). Their best, mean, and worst values are also very close to each other.

3.1. Varied Parameters Setting. In order to further verify the performance of our IPSO, we also carried out experiments on the different values of parameters and compared the experimental results with the traditional PSO. There are three parameters ω, η_1 and η_2 in the equation of the velocity update. Among them, η_1 andη_2 are the accelerating factors, and their value is constant. Generally, their value is set as $\eta 1 = \eta 2 = 2$ [1]. We will not vary them here. We just change the value of inertia weight ω. It is used to balance the global and local search capabilities. The larger the inertia weight ω, both our IPSO and the standard PSO promote global search. Note that when we vary the inertia weight ω, the values for η_1 and η_2 are varied relatively.

Step 1: randomly generate particles and give the velocity and location of each particle.
Step 2: evaluate each particle using the fitness value.
Step 3: for each particle,
If the fitness value of the particle is less than the best particle's fitness value of P_{idb}, update its location with P_{idb}.
Otherwise, if the fitness value of the particle is greater than the worst particle's fitness value P_{idw}, update its location P_{idw}.
Step 4: for each particle,
If the fitness value of the particle is less than the best fitness value P_{gdb} of the entire swarm, update the value of P_{gdb} using this particle's fitness value.
Otherwise, if the fitness value of the particle is greater than the worst fitness value P_{gdb} of the entire swarm, update the worst fitness value P_{gdw} of the entire swarm using this particle's fitness value.
Step 5: for each particle,
(1) Generate this particle's active flight direction t using (1) and (2).
(2) Generate this particle's passive flight direction $t\prime$ using (13) and (14).
(3) Compare t and t', and choose the best of them to update this particle.
Step 6: repeat Step 2 until you find the optimal solution or satisfy the termination condition.

Algorithm 1: The framework of IPSO (searching the minimum value).

Table 4: The convergence rate of PSO and IPSO for the 10 benchmark functions.

Function	Algorithm	Convergence times	Optimal solution
F1	PSO	75	1.0000000
	IPSO	100	1.0000000
F2	PSO	80	−186.730909
	IPSO	100	−186.730909
F3	PSO	90	−176.541793
	IPSO	97	−176.541793
F4	PSO	56	−1.031628
	IPSO	65	−1.031628
F5	PSO	25	3.000000
	IPSO	38	3.000000
F6	PSO	5	−39.944506
	IPSO	18	−39.944506
F7	PSO	95	0.000000
	IPSO	100	0.000000
F8	PSO	95	0.000000
	IPSO	100	0.000000
F9	PSO	100	0.000000
	IPSO	100	0.000000
F10	PSO	98	0.000000
	IPSO	100	0.000000

Table 5: The experimental results of IPSO and PSO on three multimodal functions.

Function	Algorithm	Best value	Mean value	Worst value	$f_{\min}$
F11	PSO	1495.71	4224.78	7032.89	0
	IPSO	$4.14E-29$	$4.46E-26$	$1.09E-24$	0
F12	PSO	72.51	101.41	123.95	0
	IPSO	2.19E-12	0.01	8.63E-25	0
F13	PSO	−5038.62	−4005.02	−3233.13	−12569.5
	IPSO	−9535.19	−8741.27	−8203.56	−12569.5

We show the experimental results of different values of the inertia weight from 0.1 to 0.9 on the function F3 in Table 6. Like previous experiments, each algorithm runs 100 times with the firmed accelerating factor setting $\eta 1 = \eta 2 = 2$.

Table 6 shows that our IPSO always performs better than the traditional PSO algorithm under different settings. It also shows that both our IPSO and the traditional PSO perform the best when the inertia weight $\omega = 0.6$ or 0.7. With the increase of inertia weight ω, the performance of the two algorithms is getting better. However, when the inertia weight is set not less than 0.8, our IPSO keeps its performance, but the traditional PSO gets worse with the increment of the inertia weight.

Table 6: The convergence rate of the traditional PSO and IPSO on the function F3.

ω	Algorithm	Convergence times	Optimal solution
0.1	PSO	43	−176.54793
	IPSO	50	−176.54793
0.2	PSO	56	−176.54793
	IPSO	65	−176.54793
0.3	PSO	65	−176.54793
	IPSO	73	−176.54793
0.4	PSO	75	−176.54793
	IPSO	85	−176.54793
0.5	PSO	75	−176.54793
	IPSO	85	−176.54793
0.6	PSO	90	−176.54793
	IPSO	97	−176.54793
0.7	PSO	90	−176.54793
	IPSO	97	−176.54793
0.8	PSO	85	−176.54793
	IPSO	90	−176.54793
0.9	PSO	80	−176.54793
	IPSO	90	−176.54793

4. Conclusions

An information sharing based PSO algorithm called IPSO is proposed in this paper. Comparing with the traditional PSO algorithm, improvements have been made in two areas:

First, it introduces a new information sharing mechanism: each particle's worst location and the entire swarm's worst location are also recorded. This new mechanism causes the particles to move in the opposite direction of the worst particle's locations and the worst swarm's locations. In this way, the global search space can be expanded and the probability of particles falling into local optimum can be reduced.

Second, it introduces a particle competitive strategy. This competitive strategy prompts the worst information to compete with the best. This makes the algorithm not only reduce the probability of falling into the local optimum solution, but also increase the probability of converging to the optimal solution, and the time complexity is only double that of the traditional PSO algorithm.

In summary, in this paper, we first investigated the performance of PSO by comparing it with GA and verified that PSO performs much better than GA. Then, we analyzed the shortcoming of the standard PSO. With the guidance of the analysis, we proposed a new PSO algorithm (IPSO). Our experimental results showed that IPSO performs better than the traditional PSO and the GA algorithm on benchmark functions. Especially for difficult functions, our IPSO significantly improves the success rate of finding the optimal solution.

Data Availability

The benchmark function data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (U1911205).

References

[1] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, Springer, Berlin, Germany, 2011.

[2] M. Clerc, *Particle Swarm Optimization*, John Wiley & Sons, Hoboken, NJ, USA, 2010.

[3] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[4] W. Zhang, W. Hou, C. Li, W. Yang, and M. Gen, "Multidirection update-based multiobjective particle swarm optimization for mixed No-idle flow-shop scheduling problem," *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 176–197, 2021.

[5] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, no. 1, pp. 33–57, 2007.

[6] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: a brief survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 2, pp. 262–267, 2011.

[7] J. B. Jong-Bae Park, Y. W. Yun-Won Jeong, J. R. Joong-Rin Shin, and K. Y. Lee, "An improved particle swarm optimization for nonconvex economic dispatch problems," *IEEE Transactions on Power Systems*, vol. 25, no. 1, pp. 156–166, 2010.

[8] K. Ishaque, Z. Salam, M. Amjad, and S. Mekhilef, "An improved particle swarm optimization (PSO)-based MPPT for PV with reduced steady-state oscillation," *IEEE Transactions on Power Electronics*, vol. 27, no. 8, pp. 3627–3638, 2012.

[9] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[10] A. Khare and S. Rangnekar, "A review of particle swarm optimization and its applications in solar photovoltaic system," *Applied Soft Computing*, vol. 13, no. 5, pp. 2997–3006, 2013.

[11] G.-G. Wang, A. Hossein Gandomi, X.-S. Yang, and A. Hossein Alavi, "A novel improved accelerated particle swarm optimization algorithm for global numerical optimization," *Engineering Computations*, vol. 31, no. 7, pp. 1198–1220, 2014.

[12] X. Yan, J. Gong, and Q. Wu, "Pollution source intelligent location algorithm in water quality sensor networks," *Neural Computing & Applications*, vol. 33, no. 1, pp. 209–222, 2021.

[13] Q. Wu, Z. Zhu, X. Yan, and W. Gong, "An improved particle swarm optimization algorithm for AVO elastic parameter inversion problem," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 9, pp. 1–16, 2019.

[14] X. Yan, Z. Zhou, C. Hu, and W. Gong, "Real-time location algorithms of drinking water pollution sources based on domain knowledge," *Environmental Science and Pollution Research*, vol. 28, no. 34, pp. 46266–46280, 2021.

[15] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, "A hybrid particle swarm optimization algorithm using adaptive learning strategy," *Information Sciences*, vol. 436-437, pp. 162–177, 2018.

[16] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm

optimization based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 1, pp. 82–97, 2014.

[17] W. Gong, Z. Liao, X. Mi, L. Wang, and Y. Guo, "Nonlinear equations solving with intelligent optimization algorithms: a survey," *Complex System Modeling and Simulation*, vol. 1, no. 1, pp. 15–32, 2021.

[18] W. Wang Hu and G. G. Yen, "Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 1, pp. 1–18, 2015.

[19] L. Wang, H. Geng, P. Liu et al., "Particle swarm optimization based dictionary learning for remote sensing big data," *Knowledge-Based Systems*, vol. 79, pp. 43–50, 2015.

[20] M. Mahi, Ö. K. Baykan, and H. Kodaz, "A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem," *Applied Soft Computing*, vol. 30, pp. 484–490, 2015.

[21] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and S. Deb, "A hybrid method based on krill herd and quantum-behaved particle swarm optimization," *Neural Computing & Applications*, vol. 27, no. 4, pp. 989–1006, 2016.

[22] Z. Chen, R. Xiong, and J. Cao, "Particle swarm optimization-based optimal power management of plug-in hybrid electric vehicles considering uncertain driving conditions," *Energy*, vol. 96, pp. 197–208, 2016.

[23] J. Liu, Y. Mei, and X. Li, "An analysis of the inertia weight parameter for binary particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 666–681, 2016.

[24] N. Delgarm, B. Sajadi, F. Kowsary, and S. Delgarm, "Multi-objective optimization of the building energy performance: a simulation-based approach by means of particle swarm optimization (PSO)," *Applied Energy*, vol. 170, pp. 293–303, 2016.

[25] Q. Wu, H. Liu, and X. Yan, "Multi-label classification algorithm research based on swarm intelligence," *Cluster Computing*, vol. 19, no. 4, pp. 2075–2085, 2016.

[26] S. M. Mousavi, A. Bahreininejad, S. N. Musa, and F. Yusof, "A modified particle swarm optimization for solving the integrated location and inventory control problems in a two-echelon supply chain network," *Journal of Intelligent Manufacturing*, vol. 28, no. 1, pp. 191–206, 2017.

[27] S. Chatterjee, S. Sarkar, S. Hore, N. Dey, A. S. Ashour, and V. E. Balas, "Particle swarm optimization trained neural network for structural failure prediction of multistoried RC buildings," *Neural Computing & Applications*, vol. 28, no. 8, pp. 2005–2016, 2017.

[28] W.-B. Du, W. Ying, G. Yan, Y.-B. Zhu, and X.-B. Cao, "Heterogeneous strategy particle swarm optimization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 467–471, 2017.

[29] E. Ozcan and C. K. Mohan, "Analysis of a simple particle swarm optimization system," *Intelligent Engineering Systems through Artificial Neural Networks*, vol. 8, pp. 253–258, 1998.