Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 749256, 11 pages
http://dx.doi.org/10.1155/2013/749256

Research Article

Predatory Search Strategy Based on Swarm Intelligence for Continuous Optimization Problems

J. W. Wang,[1,2,3] H. F. Wang,[4] W. H. Ip,[5] K. Furuta,[3] T. Kanno,[3] and W. J. Zhang[2]

[1] *Complex Systems Research Center, East China University of Science and Technology, Shanghai 200237, China*
[2] *Department of Mechanical Engineering, University of Saskatchewan, Saskatoon, Canada S7N 5A9*
[3] *Department of Systems Innovation, the University of Tokyo, Tokyo 113-8656, Japan*
[4] *Institute of Systems Engineering, Northeastern University, Shenyang 110114, China*
[5] *Department of Industrial and Systems Engineering, Hong Kong Polytechnic University, Kowloon, Hong Kong*

Correspondence should be addressed to W. J. Zhang; chris.zhang@usask.ca

Received 12 February 2013; Accepted 10 March 2013

Academic Editor: Zhuming Bi

We propose an approach to solve continuous variable optimization problems. The approach is based on the integration of predatory search strategy (PSS) and swarm intelligence technique. The integration is further based on two newly defined concepts proposed for the PSS, namely, "restriction" and "neighborhood," and takes the particle swarm optimization (PSO) algorithm as the local optimizer. The PSS is for the switch of exploitation and exploration (in particular by the adjustment of neighborhood), while the swarm intelligence technique is for searching the neighborhood. The proposed approach is thus named PSS-PSO. Five benchmarks are taken as test functions (including both unimodal and multimodal ones) to examine the effectiveness of the PSS-PSO with the seven well-known algorithms. The result of the test shows that the proposed approach PSS-PSO is superior to all the seven algorithms.

1. Introduction

As one of the key foundations of intelligent systems, computational intelligence is a strong tool to deal with challenging optimization problems. Most intelligent optimization methods are inspired from the biological system, such as Genetic algorithm [1], artificial immune systems [2], artificial neural network [3], ant colony optimization [4], particle swarm optimization [5, 6], culture algorithm [7], and colony location algorithm [8]. There are also methods learned from the artificial or man-made system, for example, simulated annealing [9] and fractal geometry [10]. Compared with the traditional optimization algorithms, the intelligent optimization algorithms have fewer restrictions to the objective function and are of computational efficiency and thus are widely used in industrial and social problems [11, 12].

However, to many real-world applications, the intelligent algorithms as mentioned earlier all encounter a common problem; that is, it is difficult to deal with the balance between exploitation and exploration. Exploration is the ability to test various regions in the problem space in order to locate a good optimum, hopefully the global one, and exploitation is the ability to concentrate on the search around a promising candidate solution in order to locate the optimum precisely [13]. Fast convergence velocity tends to result in the premature solution as opposed to the best solution.

Inspired by the predatory behavior of animals, Linhares [14, 15] proposed the predatory search strategy (PSS) to solve the previous problem; in particular, he used PSS to solve discrete variable optimization problems. This strategy is an individual-based searching strategy and has a good balance between exploitation and exploration.

On a general note, the balance of exploitation and exploration is not only a key problem for discrete variable optimization problems but also for the continuous variable optimization problems. A natural idea extending Linhares's work is to apply the PSS to the continuous variable optimization problem. However, the feature of individual-based searching does not allow the PSS to be straightforwardly applied to the continuous variable optimization problem. The basic motivation of this paper is to address this issue.

In this paper, we attempt to integrate the PSS with swarm intelligence techniques to solve continuous variable optimization problems. There are two major models in swarm intelligence, particle swarm optimization (PSO) and ant colony optimization (ACO). ACO is used for discrete problems, while the PSO is used for continuous problems. Therefore, in this paper, PSO is adopted to be incorporated with the PSS. The proposed approach may be called PSS-PSO. The PSS-PSO approach is tested by five benchmark examples of complex nonlinear functions. Compared with well-known PSO algorithms, the PSS-PSO is found to achieve a superior performance to the existing algorithms for continuous variable optimization problems.

In the following, we will first present the related works of PSS and PSO algorithms. Then, we will propose the basic idea of PSS-PSO and discuss the implementation of the PSS-PSO approach in Section 3, followed by the experiment and analysis in Section 4. Finally, there is a conclusion with discussion of future work.

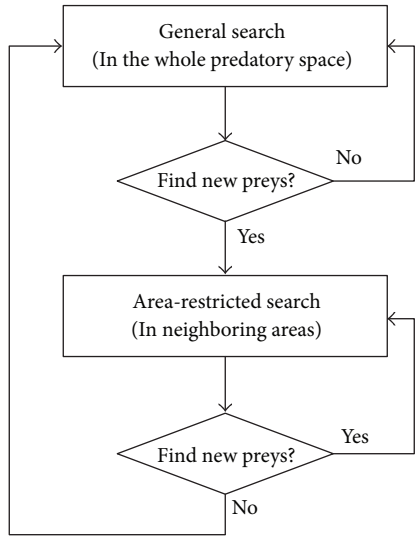

Figure 1: The predatory search strategy.

2. Related Studies

2.1. Predatory Search Strategy for Discrete Problems. Biologists have discovered that despite their various body constructions, the predatory search strategies of many search-intensive animals are amazingly similar [16–20]. When they seek food, they first search in a certain direction at a fast pace until they find a prey or sufficient evidence of it. Then, they slow down and intensify their search in the neighboring area in order to find more preys. After some time without success, they give up the intensified "area-restricted" search and go on to scan other areas [15]. As shown in Figure 1, this predatory search strategy of animals can be summarized by two search processes [21]: Search 1 (general search)—extensive search in the whole predatory space, and if prey or its proof is found, turn to search 2; Search 2 (area-restricted search)—intensive search in its neighboring area, and after a long time without success, turn to search 1.

This strategy is effective for many species because (besides being simple and general) it is able to strike a good balance between the exploitation (intensive search in a defined area) and the exploration (extensive search through many areas) of the search space. Linhares [14, 15] introduced the predatory search strategy as a new evolutionary computing technique to solve discrete optimization problems, such as TSP and VLSI layout. When this technique searches for the optimum, it first seeks solutions in the whole solution space until finding a "good" solution. Then, it intensifies the search in the vicinity of the "good" solution. After some time without further improvement, it gives up the area-restricted search and turns to the original extensive search. In this technique, a concept called *restriction* was defined as the solution cost to represent the neighborhood of a solution, which is used to adjust the neighboring area and implement the balance of exploitation and exploration.

The previous approach, in essence, can be viewed as a strategy to balance exploitation and exploration; however, it does not give any detail of the general search and area-restricted search. Hence, in this paper, we call it predatory search strategy (PSS), instead of predatory search algorithm. As mentioned in Section 1, the balance between local search and global search or exploration and exploitation is a key to all the intelligent algorithms. PSS is an example strategy to address this key. Liu and Wang [21] proposed another way to implement the PSS by adopting the solution distance as the restriction. All these studies focus on the discrete variable optimization problem.

On a general note, the general search enables the approach with a high search quality to avoid falling into the local optimum, while the area-restricted search enables the approach to have a fast convergence velocity. It is interesting to note that such search strategy was taken three decades ago in the M. S. degree thesis of Zhang [22]. In his approach, a random search technique and a gradient-based technique were combined to solve a specific continuous variable optimization problem for design of special mechanisms [22]. In his combined approach, the switch criterion for the two search algorithms was specifically related to the particular application problem; in particular, in the general search, the stop criterion is the feasible solutions in a highly constrained region [22]. The very motivation of the Zhang's combined approach was to overcome the difficulty in obtaining a feasible solution, because the constrained area was extremely "narrow." Zhang's work was at times prior to the development of intelligent optimization ideas, concepts, and techniques such as PSO. His work is inspiring but not a generalized one.

2.2. Particle Swarm Optimization and Its Developments. The PSO algorithm was first developed by Kennedy and Eberhart based on the simulation of a simplified social model [5, 6, 23]. The standard PSO algorithm can be explained as follows.

A swarm being made up by m particles searches a D-dimensional problem space. Each particle is assigned a randomized velocity and a stochastic position. The position represents the solution of the problem. When "flying" each particle is attracted by a good location achieved so far by itself

and by a good location achieved by the members in the whole swarm (or the members in the neighborhood). The position of the ith particle is represented as $x_i = (x_{i1}, \ldots, x_{id} \ldots, x_{iD})$, and its velocity is represented as $v_i = (v_{i1}, \ldots, v_{id}, \ldots, v_{iD})$, $1 \le i \le m$, $1 \le d \le D$. The best previous position of the ith particle, namely, the position with the best fitness value, is represented as $p_i = (p_{i1}, \ldots, p_{id}, \ldots, p_{iD})$, and the index of the best particle among all the particles in the neighborhood is represented by the symbol g. Each particle updates its velocity and position according to the following equations:

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 \xi \left(p_{id}^{k} - x_{id}^{k}\right) + c_2 \eta \left(p_{gd}^{k} - x_{id}^{k}\right), \quad x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1}, \tag{1}$$

where ω is the inertia weight that determines how much a particle holds its current velocity in the next iteration. c_1 and c_2 are learning factors, also named acceleration constants, which are two positive constants. Learning factors are usually equal to 2, while other settings can also be seen in the literature [13]. ξ and η are pseudorandom numbers and they obey the same homogeneous distribution in the range $[0, 1]$. The velocity of a particle is limited in the range of $V_{\max}$. $V_{\max}$ is set to be the range of each dimension variable and used to initialize the velocity of particles without selecting and tuning in detail. The running of the PSO is much similar to evolutionary algorithms such as GA, including initialization, fitness evaluation, update of velocity and position, and testing of the stop criterion. When the neighborhood of a particle is the whole population, it is called the global version of PSO (GPSO) [24]; otherwise, it is called local version of PSO (LPSO) [25].

The PSO algorithm has increasingly attracted attention now [13, 24, 26, 27] with many successful applications in real-world optimization problems [28–33]. The most salient advantage of PSO is its fast convergence to the optimum; however, this feature also tends to lead the whole swarm into the local optimum, especially for solving multimodal problems [34].

Different strategies have been developed in the literature to solve the problem with PSO. Liang et al. [34] proposed a comprehensive-learning PSO (CLPSO) which uses a novel learning strategy where the historical best information of all other particles is used to update a particle's velocity for multimodal applications. Shubham et al. [35] proposed a fuzzy clustering-based particle swarm (FCPSO) algorithm by using an external repository to preserve nondominated particles found along the search process to solve multiobjective optimization problems. Zhan et al. [36] proposed an adaptive PSO (APSO) by developing a systematic parameter adaptation scheme and an elitist learning strategy. Clerc [37] proposed an adaptive PSO, called TRIBES, in which the parameters change according to the swarm behavior. TRIBES is a totally parameter free algorithm and the users only modify the adaptive rules [37, 38]. In the work of [39], the flight mechanism of wild goose team, including goose role division, parallel principle, aggregate principle, and separate principle, was proposed to address the problem with PSO [39]. Silva et al. [40] proposed a variant of PSO, called predatory prey optimizer (PPO), by introducing another swarm; in particular, original swarm is called prey swarm and the new swarm is called predator swarm. These two swarms have different dynamic behaviors which will balance the exploration and exploitation in PSO. They further proposed a simple adaptable PPO by introducing a symbiosis adaptive scheme into PPO [41]. The difference between their work and ours will be discussed in the next section after some detail of our approach is introduced.

3. The Proposed PSS-PSO Approach

3.1. Basic Idea. To make the PSS be capable of solving continuous variable optimization problems, we need to address three issues: (1) how to search the continuous space, including the general search and the area-restricted search, (2) how to define restrictions, and (3) how to define the neighborhood. The first issue can be addressed by the PSO; so we will not discuss it here. In the following, we will address the last two issues. We start with proposing two concepts, namely *Restriction* and *Neighborhood*.

Restriction. It is the distance between two points in a multidimensional space. Here, we use the Euclid norm to define the distance.

Neighborhood. The neighborhood of a point, or a particle, under some restriction, is defined as a hyperspherical space which takes the particle as the center and the restriction as the radius.

Further, there are Lrestrictions in the initial solution space, which are represented as $restriction(0), \ldots, restriction(l), \ldots, restriction(L-1)$, where $l \in \{0, 1, \ldots, L\}$ is called the level of the restriction. Therefore, the neighborhood of a point, say x, under the restriction, $restriction(l)$, can be represented as $N(x, restriction(l))$.

The list of the restriction levels needs to be recomputed when a new best overall solution, b, is found, because a restricted search area should be defined around this new improving point. To build such a restriction list, the following scheme is used: initialize L particles in the initialization space and compute each distance between one particle and b. Rank the Ldistances. It is noted that $restriction(0)$ carries the smallest distance, and $restriction(1)$ carries the second smallest distance, and so on.

The overall procedure of PSS is given as follows.

Choose one random point x in the initial space ψ. A swarm consisting of m particles is initialized in $N(x, restriction(0))$, and then the search is carried out with the standard PSO update equations. The number of iterations is controlled by variable n. If no better solution can be found, the swarm will be initialized in $N(x, restriction(1))$, and repeat the search; if a better solution, b, is found, then the best solution will be updated by this new solution, and all the restrictions will be set up again. The neighborhood is enlarged or reduced through the adjustment of restrictions, which makes sense for the concept of balancing exploration and exploitation.

```
Begin
Choose x randomly from ψ
while l < L
        initialize m particles in N (x, restriction(l))
        for (i = 0; i < I; i + +)
                for (c = 0; c < n; c + +)
                        update swarm based on standard PSO equations
                        if g < p then p = g
                end for
                x = p
                if f(x) < f(b) then
                b = x, l = 0, i = 0, re-compute L restrictions
        end for
        l = l + 1
        if l = [L/5] then
                l = L − [L/S]
End while
```

Algorithm 1: Pseudocode of the PSS-PSO algorithm.

As mentioned in Section 2.2, Silva et al. [40, 41] proposed algorithms with similar names of ours. However, our idea is totally different from theirs. In their work, they used two swarms in PSO and each swarm has different behavior. The particles in the prey swarm have the same behavior with the standard PSO; the particles in the predatory swarm are attracted to the best individuals in the swarm, while the other particles are repelled by their presence [40, 41]. The essence of such a mechanism with their approach is a multipopulation strategy to keep the diversity of the whole swarm. Our idea only considers the behavior of the predator and the balance of exploration and exploitation is achieved by the movement of predatory individuals not a predatory swarm. Furthermore, there is only one swarm performing the local search in our algorithm PSS-PSO, while there are two swarms in their algorithm PPO.

3.2. Implementation. The pseudocode of the PSS-PSO algorithm is presented in Algorithm 1.

Let

ψ: The initial space;

L: The number of restrictions;

l: The level of the restriction, $l \in \{0, 1, \ldots, L\}$;

n: The maximum number of iterations of PSO;

c: The iteration number of PSO;

I: The maximum running times of PSO;

i: The running number of PSO;

g: Best solution of PSO in each iteration;

m: Number of particles in a swarm;

$N(x, restriction(l))$: The neighborhood of x, under the restriction restriction(l), $l \in \{0, 1, \ldots, L\}$;

b: A new best overall solution in ψ;

p: The best solution in $N(x, restriction(l))$.

It is noted that we used the integer division of L as the number of restriction levels for each search mode. The algorithm was made to stop when $[L/5]$ levels in general search are tried without finding a new better solution.

4. Experiments

4.1. Experimental Setting. Five benchmarks [13, 24, 42] which are commonly used in the evolutionary computation are selected as examples in this paper to test the proposed approach. The detail of the functions, including the formulations, the numbers of dimensions, and the search range of the variables, are summarized in Table 1. It is noted that in the experiment, the initial range of algorithm is set to be the same with the search range. The Sphere function and the Rosenbrock function are two unimodal functions and have only one global minimum in the initial search space. The Rosenbrock function changes very slightly in the long and narrow area close to the global minimum. The Schaffer's f_6 function, the Rastrigrin functions, and the Griewank function are three multimodal functions, which have very complicated landforms. The global optimums of the five functions are all zero. The five functions are quite fit for the evaluation of algorithm performance [42].

Seven existing PSO algorithms, as listed in Table 2, are compared with PSS-PSO. First three PSO algorithms, including GPSO, LPSO with ring neighborhood, and VPSO with von Neumann neighborhood, are well known and have widely been used in computation comparisons and real-world applications. The other three PSO algorithms are recent versions as mentioned in Section 2.2. CLPSO adopts a novel learning strategy and aims at better performance for multimodal functions. GTO refers to the goose team flight mechanism and the gradient information of functions and obtains fast convergence for the unimodal functions and some multimodal functions. APSO offers a systematic parameter adaptation scheme which can speed up other PSO variants. TRIBES

Table 1: Benchmarks.

Function	Formula	Dim	Search range
Sphere	$f_1(\vec{x}) = \sum_{i=1}^{n} x_i^2$	30	$[-100, 100]^n$
Rosenbrock	$f_2(\vec{x}) = \sum_{i=1}^{n-1} \left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$	30	$[-30, 30]^n$
Schaffer's f_6	$f_3(\vec{x}) = 0.5 + \frac{\left(\sin\sqrt{x_1^2 + x_2^2}\right)^2 - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	2	$[-100, 100]^2$
Rastrigrin	$f_4(\vec{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	30	$[-5.12, 5.12]^n$
Griewank	$f_5(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	$[-600, 600]^n$

Table 2: PSO algorithms used in the comparison.

Algorithm	Reference
GPSO	Shi and Eberhart, 1998 [24]
LPSO	Kennedy and Mendes, 2002 [25]
VPSO	Kennedy and Mendes, 2002 [25]
CLPSO	Liang et al., 2006 [34]
GTO	J. Wang and D. Wang, 2008 [39]
APSO	Zhan et al., 2009 [36]
TRIBES	Cooren et al., 2009 [38]

is an adaptive algorithm which enables the users to be free in the parameter selection.

First, we will compare the PSS-PSO with the first six PSO algorithms on the ability of exploitation and exploration with the five benchmarks listed in Table 1. Then, we will compare the PSS-PSO with TRIBES on the four shifted benchmarks. GPSO, GTO, and PSS are programmed by Java run on a PC with P4-2.66 GHz CPU and 512 MB memory. Results of the other algorithms are from the literature, respectively. To get rid of the randomicity, the results are the average of 30 trial runs [42]. To test exploitation and exploration, we use different swarm sizes and stop criterions, which are given in the following discussion.

4.2. Comparisons on the Exploitation

4.2.1. Comparisons with Different Algorithms. For a fair comparison, the PSS and the first six PSO algorithms are tested with the same population size (m) of 20. The stop criterion is set according to reference [42] as number of function evaluations (FEs) = 200000. With consideration of the population size, this stop criterion also means that the number of iteration is 10000. In the following discussion, we will use the iteration number as the stop criterion. Here, we define the global optimum zero as "less than 10^{-45}." If the algorithm reaches the global optimum, it also ends. It is noted that, similar to the PSS for discrete problem, the PSS-PSO for continuous problems uses the restriction level as the stop criterion (see Algorithm 1). To compare with other PSO algorithms, we also record the iteration number and stop PSS-PSO when the iteration number reaches 10000.

The average optimums obtained by the seven algorithms are listed in Table 3. The data of PSS and GTO, are obtained by our experiments. Other data are from the references listed in Table 2. Symbol "—" in Table 3 means the data unavailable. It is noted that the results of CLPSO and APSO on the Rosenbrock function are with symbol "∗". The reason is that the two algorithms are not good at dealing with unimodal functions, particularly with the Rosenbrock functions. The initial ranges of the Rosenbrock functions for CLPSO and APSO are listed in Table 4 [34, 36], and therefore, the common idea is to narrow the search space of this function.

On the simple unimodal function Sphere, PSS-PSO, GPSO, GTO and APSO can all find the global optimum. Actually, GTO is the fastest one, which will be shown from the comparison on the exploration in the next section. On the complicated unimodal function Rosenbrock, PSS-PSO is the best one. Although GTO references the gradient information in the optimization process, the complicated search space still prevents it from obtaining the satisfactory result. On the simple multimodal function Schaffer's f_6, PSS-PSO, GPSO, and GTO can all reach the global optimum. On the complicated multimodal functions, Rastrigrin and Griewank, PSS-PSO, CLSPO and APSO are all good optimizers. PSS-PSO and CLPSO have better performance on Griewank; CLSPO and APSO have better performance on Rastrigrin. The results prove the theorem of "no free lunch" that no algorithm can outperform all the others on every aspects or every kind of problems [43]. However, the PSS-PSO outperforms the six PSO algorithms on most of the benchmark functions except the Rastrigrin function.

4.2.2. Analysis of the Search Behaviors. To further observe the search behaviors of the PSS-PSO, we try other two population sizes, 10 and 30, respectively, and extend the maximum iteration numbers to 50000 in the experiments and give the average convergence curves of PSS-PSO and GPSO on the five functions in Figure 2. In this figure, the x coordinate is the iteration number. The y coordinate is $\lg f(x)$, the common logarithm of the fitness value, because the fitness value changes too much in the optimization process.

(1) Exploitation for Unimodal Functions. From the convergence curves on the Sphere function, we can see that GPSO has a faster convergence speed. The reason is that Sphere

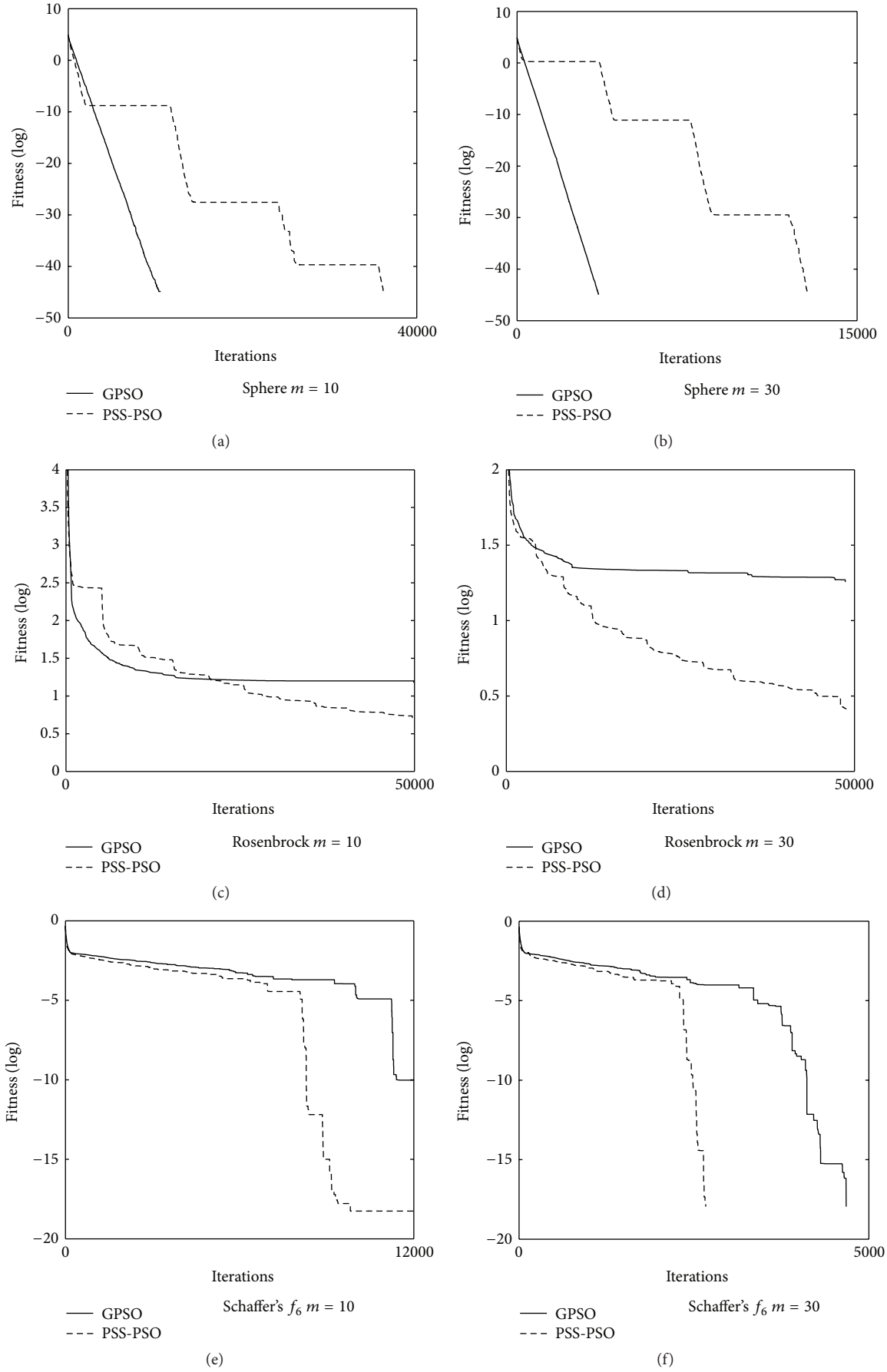

Figure 2: Continued.

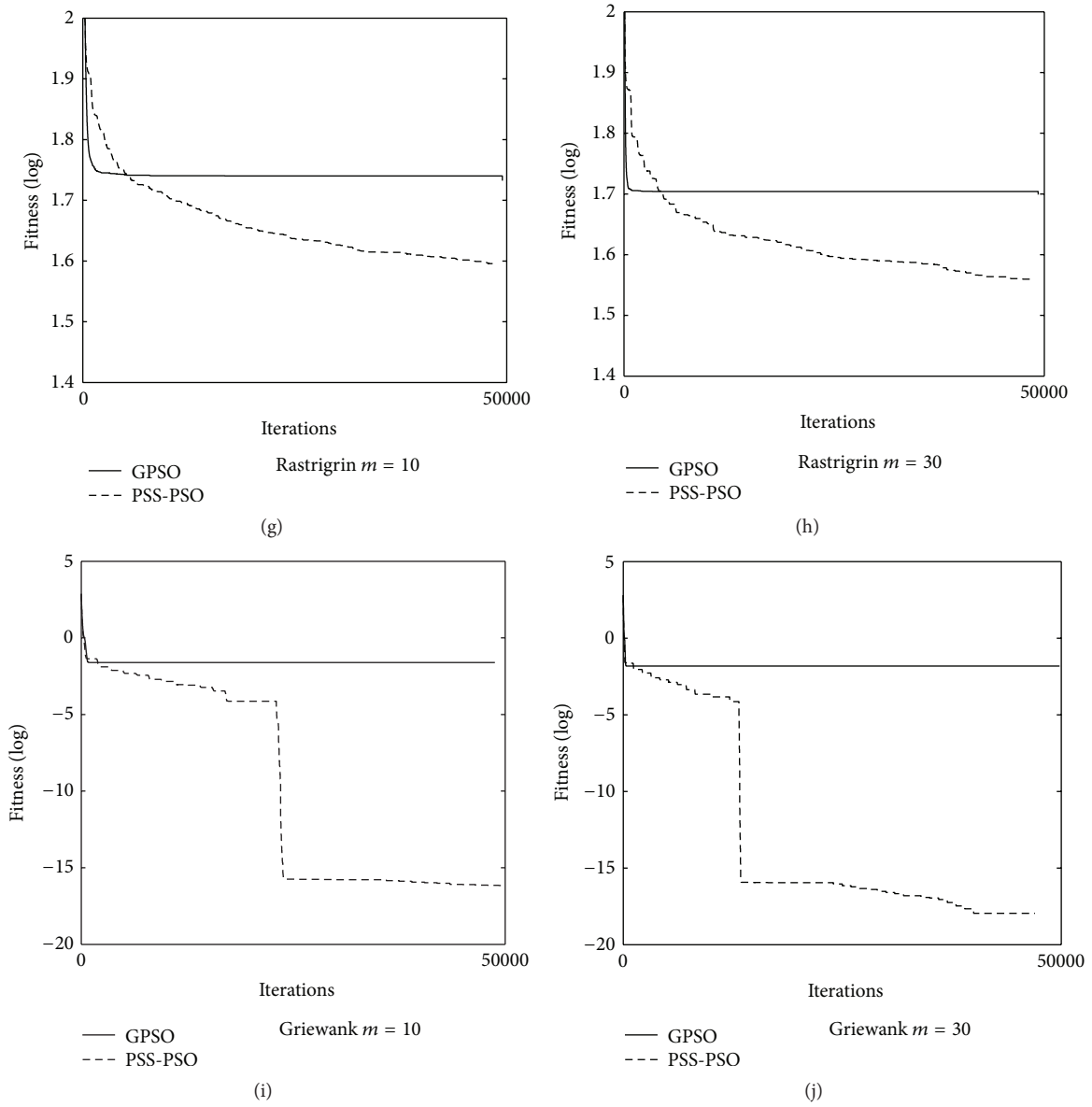

Figure 2: Average convergence curves of GPSO and PSS-PSO over 30 trials.

Table 3: Average optimums of seven algorithms.

Algorithm	Sphere	Rosenbrock	Schaffer's f_6	Rastrigrin	Griewank
PSS-PSO	0	0.0813065	0	13.2057	$1.0214E-16$
GPSO	0	24.486	0	29.6099	$2.37E-2$
LPSO	$6.33E-29$	18.9472	—	35.0819	$1.1E-2$
VPSO	$8.49E-39$	35.9005	—	27.7394	$1.31E-2$
CLPSO	$1.06E019$	11*	—	$7.4E-11$	$6.45E-13$
GTO	0	5.1711316	0	215.24	$5.7518E-4$
APSO	0	2.84*	—	$5.8E-15$	$1.67E-2$

Table 4: Search ranges of Rosenbrock for CLPSO and APSO.

Algorithm	Search range
CLPSO	$[-2.048, 2.048]^n$
APSO	$[-10, 10]^n$

Table 5: Goal for exploration.

Function	Sphere	Rosenbrock	Schaffer's f_6	Rastrigrin	Griewank
Goal	0.01	100	10^{-5}	100	0.1

is a simple unimodal function and the particles easily find the global optimum. On the optimization of the Rosenbrock functions, the problem of GPSO appears (i.e., not having a sustaining search ability during long iterations); in particular, GPSO cannot find better solutions after about 10000 iterations. On the contrary, PSS-PSO can obtain better solutions during the whole optimization process. The reason is that, as mentioned in Section 4.1, the Rosenbrock function changes very slightly in the long and narrow area close to the global minimum. This feature makes GPSO lose its ability of dynamic adjustment of particle's velocity, which leads the whole swarm to lose the energy of reaching better solutions. However, the switch of different restriction levels enables the swarm in the PSS-PSO with its ability of dynamic adjustment of particle's velocity continuously.

(2) Exploitation for Multimodal Functions. On the simple multimodal function Schaffer's f_6, GPSO and PSS-PSO can both perform well. However, on the two complicated functions, Rastrigrin and Griewank, GPSO exposed its deficiency again. After very early iterations, the curve of GPSO stays horizontal, which means that the whole swarm has been trapped into a local optimum. However, PSS-PSO does not have this problem, and the curves of PSS-PSO go down continuously.

4.3. Comparisons on the Exploration. To test the exploration of PSS-PSO, we use the following criterion in the experiment: whether the algorithm can find the goal set for each function in 4000 iterations. Using this criterion, we can get success rate, average iteration number, and average running time. Here, average iteration number is the iteration number for the successful runs; success rate is defined as number of successful runs/total number of runs. The predefined accuracy levels, namely, the goals of the different functions, are listed in Table 5. Since exploration is the ability to test various regions in the problem space in order to locate a good optimum, what we care about in this experiments is the convergence speed. Therefore, the goal value here is different from the global optimum, and in particular, the goal is an acceptable good solution. We compare PSS-PSO with GPSO and GTO in the experiments. We try two population sizes, 10 and 30, respectively, and run the algorithms 100 times for each case. The results are listed in Table 6.

The results show that even with a very small population size 10, PSS can still achieve 100% success rates for all the functions. As the standard version of PSO, GPSO has the worst performance, which is not beyond our expectation. On the three functions, Sphere, Schaffer's f_6, and Griewank, GTO can reach the goals very fast in even 1 or 3 iterations. This fast convergence speed of GTO benefits from the gradient information. However, on the two complicated functions, Rosenbrock and Rastrigrin, GTO does not have such good performance. It even fails to reach the goal of the Rastrigrin function at any trial. The comparison shows that PSS-PSO has an overall good performance on the exploration.

4.4. Comparisons on the Shifted Benchmarks. In the previous experiments, all the benchmarks are traditional functions with the global optimum at the origin of the search space, which may be more easily for some algorithms to find the global optimum [44]. Therefore, we use the shifted versions of benchmarks in Table 1 to perform a further test; in particular, the results are compared with that of TRIBES from the literature [38]. The shifted benchmarks are defined by Suganthan et al. [42] and labeled as F1 (shifted Sphere), F6 (shifted Rosenbrock), F7 (shifted rotated Griewank without bound), F10 (shifted rotated Rastrigrin), and F14 (shifted rotated expanded Schaffer's f_6). Because TIBES can not give satisfying result of F14 [38], we did not consider F14 in this experiment. Two tests are performed to examine the exploitation and exploration of two algorithms. To examine the exploitation, the algorithm stops if the number of evaluations of the objective functions exceeds $10000 * D$. The dimension D is set as 10. Each algorithm runs 25 times for one case. Then, we recorded the mean error for each benchmark. To test the exploration, we examine the number of function evaluations the algorithms needs to achieve a goal. The algorithm stops if it reachs the goal or if the number of evaluations of the objective functions exceeds $10000 * D$. The dimension D is also set as 10. Each algorithm runs 25 times for one case. The goals of functions, the mean of the number of functions evaluations (mean iterations for short), the success rates, and the performance rates are recorded. The performance rate is defined as mean iterations/success rate. All the results are listed in Table 7.

From the results in Table 7, we can see that the PSS-PSO has similar performance with TRIBES in F1, F6, and F7. PSS-PSO and TRIBES can both find global optimum in F1, which is a unimodel function, and PSS-PSO is a little slower than TRIBES. PSS-PSO and TRIBES both have very low ability to reach the goals for F6 and F7. In the runs that find the goals, PSS-PSO needs fewer number of iterations.

However, PSS-PSO has a much better result than TRIBES in F10. TRIBES cannot reach the goal of F10; on contrary, PSS-PSO has a success rate of 0.12. Furthermore, PSS-PSO can achieve a much less mean error in F10.

5. Conclusions

In this paper, we proposed an approach to integrate PSS with PSO, and the approach was named PSS-PSO. This integration is achieved by proposing two concepts with the PSS, namely restriction and neighborhood. Five benchmarks are taken as test functions to examine the exploration and

Table 6: Exploration performance of the different algorithms.

m	Algorithm	Index	Sphere	Rosenbrock	Schaffer's f_6	Rastrigrin	Griewank
10	GPSO	Suc. rate	1	0.82	0.75	0.97	0.96
		Iterations	1379.95	1847.4512	1808.3334	605.8866	1149.5938
		Ave. time (s)	0.216	0.360	0.059	0.154	0.264
	GTO	Suc. rate	1	0.98	1	0	1
		Iterations	1.04	164.79	3.2333333	—	1.01
		Ave. time (s)	0.009	0.038	0.011	—	0.009
	PSS-PSO	Suc. rate	1	1	1	1	1
		Iterations	832.66	1879.39	1570.9	690.5	998.34
		Ave. time (s)	0.140	0.317	0.040	0.144	0.200
30	GPSO	Suc. rate	1	0.93	1	1	0.99
		Iterations	320.59	780.17896	959.46	226.89	433.21213
		Ave. time (s)	0.159	0.464	0.058	0.144	0.292
	GTO	Suc. rate	1	1	1	0	1
		Iterations	1.06	111.31	3.2666	—	1
		Ave. time (s)	0.01	0.038	0.011	—	0.01
	PSS-PSO	Suc. rate	1	1	1	1	1
		Iterations	332.81	648.07	434.0	243.94	319.57
		Ave. time (s)	0.163	0.311	0.030	0.151	0.192

Table 7: Performance on shifted benchmarks.

Index	Algorithm	F1	F6	F7	F10
Accuracy		$1e-06$	$1e-02$	$1e-02$	$1e-02$
Mean iterations	PSS-PSO	4366	77290	62600	61600
	TRIBES	1364.72	98309.28	96568.44	$1E+05$
Success. rate	PSS-PSO	1	0.04	0.04	0.12
	TRIBES	1	0..04	0.04	0
Performance rate.	PSS-PSO	4366	1932250	1565000	513333.33
	TRIBES	1364.74	2457725	2414211	—
Mean error	PSS-PSO	0	9.692547	0.05717346	1.8716955
	TRIBES	0	0.85882	0.077474	12.118002

exploitation of the proposed approach. Experiments show that the ability of balancing exploration and exploitation makes the proposed approach be applicable for both the unimodal and multimodal functions. Compared with six existing PSO algorithms, PSS-PSO has achieved a better performance overall.

Some future research needs to be carried out on the proposed approach. First, we will test the PSS-PSO for more benchmarks, especially for more rotated functions. Second, we will test the PSS-PSO in complex environments; in particular, we will use it to solve the dynamic problems and multiple objective optimization problems. Third, we will use the PSS-PSO to solve the complex real-world applications such as the mixed topology and geometry optimization design problems in robotics [45].

Acknowledgments

The authors want to thank the financial support to this work by NSERC through a strategic project grant, and a grant of ECUST through a program of the Fundamental Research Funds for the Central Universities to W. J. Zhang, and by NSFC (Grant No. 71001018, 61273031) of China to H. F Wang. Our gratitude is also extended to the Department of Industrial and Systems Engineering of the Hong Kong Polytechnic University (G-YK04).

References

[1] J. H. Holland, *Adaptation in Nature and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] J. D. Farmer, N. H. Packard, and A. S. Perelson, "The immune system, adaptation, and machine learning," *Physica D*, vol. 22, no. 1–3, pp. 187–204, 1986.

[3] J. J. Hopfield and D. W. Tank, "Neural computation of decisons in optimization problems," *Biological Cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[4] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.

[5] R. Eberhart and J. Kennedy, "New optimizer using particle swarm theory," in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pp. 39–43, Indianapolis, Ind, USA, October 1995.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.

[7] R. G. Reynolds and S. R. Rolnick, "Learning the parameters for a gradient-based approach to image segmentation from the results of a region growing approach using cultural algorithms," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 819–824, Detroit, Mich, USA, December 1995.

[8] D. Wang, "Colony location algorithm for assignment problems," *Journal of Control Theory and Applications*, vol. 2, no. 2, pp. 111–116, 2004.

[9] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[10] B. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman and Co., San Francisco, Calif, USA, 1982.

[11] H. Liu, "A fuzzy qualitative framework for connecting robot qualitative and quantitative representations," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1522–1530, 2008.

[12] R. E. Hiromoto and M. Manic, "Information-based algorithmic design of a neural network classifier," *International Scientific Journal of Computing*, vol. 5, no. 3, pp. 87–98, 2006.

[13] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, 2003.

[14] A. Linhares, "State-space search strategies gleaned from animal behavior: a traveling salesman experiment," *Biological Cybernetics*, vol. 78, no. 3, pp. 167–173, 1998.

[15] A. Linhares, "Synthesizing a predatory search strategy for VLSI layouts," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 147–152, 1999.

[16] J. N. M. Smith, "The food searching behaviour of two European thrushes. II. The adaptiveness of the search patterns," *Behavior*, vol. 49, no. 1-2, pp. 1–61, 1974.

[17] K. Nakamuta, "Mechanism of the switchover from extensive to area-concentrated search behaviour of the ladybird beetle, Coccinella septempunctata bruckii," *Journal of Insect Physiology*, vol. 31, no. 11, pp. 849–856, 1985.

[18] W. J. Bell, "Searching behavior patterns in insects," *Annual Review of Entomology*, vol. 35, pp. 447–467, 1990.

[19] P. Kareiva and G. Odell, "Swarms of predators exhibit "prey-taxis" if individual predators use area-restricted search," *American Naturalist*, vol. 130, no. 2, pp. 233–270, 1987.

[20] E. Curio, *The Ethology of Predation*, Springer, Berlin, Germany, 1976.

[21] C. Liu and D. Wang, "Predatory search algorithm with restriction of solution distance," *Biological Cybernetics*, vol. 92, no. 5, pp. 293–302, 2005.

[22] W. J. Zhang, *Studies on design of spatial crank-rocker mechanisms based on the pressure angle [M.S. thesis]*, Dong Hua University, Shanghai, China, 1984.

[23] J. Kennedy, "Particle swarm: social adaptation of knowledge," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '97)*, pp. 303–308, Washington, DC, USA, April 1997.

[24] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, Indianapolis, Ind, USA, May 1998.

[25] J. Kennedy and R. Mendes, "Population structure and particle swarm performancein," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1671–1676, Honolulu, Hawaii, USA, 2002.

[26] S.-Z. Zhao and P. N. Suganthan, "Two-lbests based multi-objective particle swarm optimizer," *Engineering Optimization*, vol. 43, no. 1, pp. 1–17, 2011.

[27] C. L. Sun, J. C. Zeng, and J. S. Pan, "An improved vector particle swarm optimization for constrained optimization problems," *Information Sciences*, vol. 181, no. 6, pp. 1153–1163, 2011.

[28] R. D. Araújo, "Swarm-based translation-invariant morphological prediction method for financial time series forecasting," *Information Sciences*, vol. 180, no. 24, pp. 4784–4805, 2010.

[29] S.-K. Oh, W.-D. Kim, W. Pedrycz, and B.-J. Park, "Polynomial-based radial basis function neural networks (P-RBF NNs) realized with the aid of particle swarm optimization," *Fuzzy Sets and Systems*, vol. 163, pp. 54–77, 2011.

[30] W. Pedrycz and K. Hirota, "Fuzzy vector quantization with the particle swarm optimization: a study in fuzzy granulation-degranulation information processing," *Signal Processing*, vol. 87, no. 9, pp. 2061–2074, 2007.

[31] W. Pedrycz, B. J. Park, and N. J. Pizzi, "Identifying core sets of discriminatory features using particle swarm optimization," *Expert Systems with Applications*, vol. 36, no. 3, pp. 4610–4616, 2009.

[32] J. W. Wang, W. H. Ip, and W. J. Zhang, "An integrated road construction and resource planning approach to the evacuation of victims from single source to multiple destinations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 2, pp. 277–289, 2010.

[33] J. W. Wang, H. F. Wang, W. J. Zhang, W. H. Ip, and K. Furuta, "Evacuation planning based on the contraflow technique with consideration of evacuation priorities and traffic setup time," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 480–485, 2013.

[34] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[35] S. Agrawal, B. K. Panigrahi, and M. K. Tiwari, "Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 5, pp. 529–541, 2008.

[36] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 39, no. 6, pp. 1362–1381, 2009.

[37] M. Clerc, "Particle swarm optimization," in *International Scientific and Technical Encyclopaedia*, Wiley, Hoboken, NJ, USA, 2006.

[38] Y. Cooren, M. Clerc, and P. Siarry, "Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm," *Swarm Intelligence*, vol. 3, no. 2, pp. 149–178, 2009.

[39] J. Wang and D. Wang, "Particle swarm optimization with a leader and followers," *Progress in Natural Science*, vol. 18, no. 11, pp. 1437–1443, 2008.

[40] A. Silva, A. Neves, and E. Costa, *An Empirical Comparison of Particle Swarm and Predator Prey Optimisation*, vol. 2464 of

Lecture Notes in Computer Science, Springer, Berlin, Germany, 2002.

[41] R. Silva, A. Neves, and E. Costa, *SAPPO: A Simple, Adaptable, Predator Prey Optimiser*, vol. 2902 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2003.

[42] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimizationin," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–50, 2005.

[43] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[44] W. Zhong, J. Liu, M. Xue, and L. Jiao, "A multiagent genetic algorithm for global numerical optimization," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 2, pp. 1128–1141, 2004.

[45] Z. M. Bi, *Adaptive robotic manufacturing systems [Ph.D. thesis]*, Department of Mechanical Engineering, University of Saskatchewan, Saskatchewan, Canada, 2002.