Hindawi Publishing Corporation
The Scientific World Journal
Volume 2014, Article ID 713490, 15 pages
http://dx.doi.org/10.1155/2014/713490

Research Article

A Particle Swarm Optimization Variant with an Inner Variable Learning Strategy

Guohua Wu,[1,2] Witold Pedrycz,[2,3,4] Manhao Ma,[1] Dishan Qiu,[1] Haifeng Li,[5] and Jin Liu[1]

[1] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha, Hunan 410073, China*
[2] *Department of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada T6R 2V4*
[3] *Warsaw School of Information Technology, Newelska, 01-447 Warsaw, Poland*
[4] *Department of Electrical and Computer Engineering, Faculty of Engineering, King Abdulaziz University, Jeddah 21589, Saudi Arabia*
[5] *School of Civil Engineering and Architecture, Central South University, Changsha, Hunan 410004, China*

Correspondence should be addressed to Haifeng Li; lihaifeng1@csu.edu.cn

Received 27 August 2013; Accepted 10 October 2013; Published 23 January 2014

Academic Editors: Y. Deng and Y. Zhao

Although Particle Swarm Optimization (PSO) has demonstrated competitive performance in solving global optimization problems, it exhibits some limitations when dealing with optimization problems with high dimensionality and complex landscape. In this paper, we integrate some problem-oriented knowledge into the design of a certain PSO variant. The resulting novel PSO algorithm with an inner variable learning strategy (PSO-IVL) is particularly efficient for optimizing functions with symmetric variables. Symmetric variables of the optimized function have to satisfy a certain quantitative relation. Based on this knowledge, the inner variable learning (IVL) strategy helps the particle to inspect the relation among its inner variables, determine the exemplar variable for all other variables, and then make each variable learn from the exemplar variable in terms of their quantitative relations. In addition, we design a new trap detection and jumping out strategy to help particles escape from local optima. The trap detection operation is employed at the level of individual particles whereas the trap jumping out strategy is adaptive in its nature. Experimental simulations completed for some representative optimization functions demonstrate the excellent performance of PSO-IVL. The effectiveness of the PSO-IVL stresses a usefulness of augmenting evolutionary algorithms by problem-oriented domain knowledge.

1. Introduction

Optimization plays an important role in scientific research, management, industry, and so forth, given the fact that many problems in the real world are essentially optimization tasks. However, with the increase of complexity of optimization problems associated with multimodality, noise, and high dimensionality of problems, "traditional" optimization methods (e.g., gradient-based methods) are no longer completely effective when searching for optimal or satisfactory solutions within the bounds of reasonable computation cost. In light of these challenges, many bioinspired algorithms, such as Genetic Algorithms (GAs) and Ant Colony Optimization (ACO), have emerged. Particle Swarm Optimization (PSO), developed by Kennedy and Eberhart [1, 2], is a competitive population-based algorithm being particularly efficient when dealing with continuous optimization problems. It is a swarm intelligence [3] algorithm that emulates swarm behaviors such as birds flocking and fish schooling [4]. Each particle in PSO adjusts its flying speed and direction by learning from its own past experience and neighbors' experience, attempting to search for better position gradually [5].

Due to its powerful capability and relatively low number of parameters, PSO has drawn wide attention since its inception. To enhance the efficiency of the generic version of the PSO method, many variants have been presented. These variants are realized through different augmentations of the generic method, generally including parameter tuning [6–11], topology structure adjustment [12–16], intelligent combination of various search strategies [17–19], and hybridization with other classical optimization techniques [20–23]. Although significant progress and achievements have been

obtained, still a fundamental challenge on how to make PSO successful in determining optimal or near optimal solutions for optimization problems with complicated landscapes and of high dimensionality still remains. In addition, even though PSO has been praised for many merits, including simple implementation, it has been criticized for suffering from premature and the quick performance degradation in case of increasing dimensionality of the optimization problem [24].

Noticeably, previous PSO variants generally focus on the modification of particle's behaviors, to strengthen simultaneously its exploration and exploitation capabilities. These efforts indeed improve significantly the effectiveness of the generic PSO. Another promising direction in improving the PSO performance is to acquire and utilize the domain knowledge associated with the optimization problems at hand. Subsequently this domain knowledge can be integrated into the search strategy in anticipation of delivering more effective search guidance for the particles. As a matter of fact, the combination of knowledge-based strategy with the heuristics of swarm optimization has been demonstrated to be effective in discrete optimization [25–27]. Note that the problem domain knowledge in discrete optimization (e.g., the scheduling problem [26, 27] and the spatial geoinformation services composition problem [28, 29]) is dependent on concrete problems considered and the knowledge extraction and discovery process is relatively subjective. In [30], the authors proposed a variable reduction strategy by utilizing the knowledge of derivative equations of unconstraint optimization problems, to reduce the complexity of original optimization problems.

The notion of variable symmetry can be encountered in optimization functions. Variable symmetry means that all or some variables encountered in the function under optimization are symmetric; namely, they can exchange positions through linear transformation without affecting the original function. We refer to such functions in which all variables are symmetric as completely symmetric function. There are functions in which only some variables are symmetric giving rise to the concept of partially symmetric function. In general, symmetric functions are developed by using operators of summation and product ("$\sum$" and "$\prod$"). According to this observation, we note that all symmetric variables in the optimal solutions of such a function are supposed to satisfy a certain quantitative relation. The domain knowledge acquired about symmetric functions becomes useful in the enhancement of the search performance. The underlying motivation of this study is to utilize such domain knowledge to strengthen the PSO's capability in solving optimization problems with symmetric variables.

The major contributions of the paper can be summarized as follows.

(1) Based on the knowledge that symmetric variables in the optimal solution of an optimization function satisfy a certain quantitative relation, we present an inner variable learning (IVL) strategy to provide particles with exact and efficient search guidance.

(2) We design a trap detection strategy, by which one can determine if the particle has been trapped in a local optimum. We also employ an adaptive Gaussian mutation-based trap jumping out strategy to help particles to escape from local optima.

(3) We propose a new knowledge-driven PSO variant, named PSO-IVL, which is integrated with the IVL strategy, trap detection and jumping out strategy, and the basic PSO.

(4) Extensive experimental simulations and analysis are conducted to demonstrate the efficiency of PSO-IVL in solving global optimization functions with symmetric variables and offer a comparison with some other state-of-the-art PSO variants.

The paper is structured as follows. Section 2 briefly introduces the basic PSO and reviews related work existing in the literature. Section 3 details the IVL strategy. Section 4 introduces the trap detection and jumping out strategy and proposes the algorithm framework of PSO-IVL. Section 5 reports experimental simulations and offers a detailed performance analysis. Section 6 concludes this paper identifying future research directions.

2. Related Studies

PSO has undergone significant progress since its introduction in 1995. A large number of PSO variants have been proposed to improve the performance of traditional PSO. Comprehensive reviews of PSO can be found in [36–38]. In addition, Valle et al. [39] surveyed PSO along with its basic concepts, variants, and applications in power systems. Rana et al. [40] reviewed PSO and its application to data clustering. In this section, we first briefly introduce the basic PSO and then survey the major PSO variants.

2.1. Basic PSO. Analogous to some other evolutionary algorithms, such as Genetic Algorithm and Ant Colony Optimization, PSO is a population-based stochastic optimization algorithm. A swarm of particles in PSO attempt to search for superior solutions through learning, communication, and interaction. The position of each particle refers to a solution. Then the position moving process of a particle in the solution space relates to a solution search process. The state of particle i is described by its current position $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ and velocity $\mathbf{v_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$, where D stands for the number of variables encountered in the optimization problem. In the generic PSO with inertia weight [31], the position and velocity of particle i are updated during the evolutionary process:

$$v'_{id} \longleftarrow w \times v_{id} + c_1 \times r_{1d} \times \left(p\text{Best}_{id} - x_{id}\right) + c_2 \times r_{2d} \times \left(g\text{Best}_{d} - x_{id}\right), \quad (1)$$

$$x'_{id} \longleftarrow x_{id} + v'_{id}, \quad (2)$$

where x'_{id} and x_{id} represent the dth variable (or dimension) of the next and current positions of particle i; v'_{id} and v_{id} denote the dth variable of the next and current velocities of particle i; $p\text{Best}_{id}$ is the dth variable of the personal historical

best position found by particle i up to now, and $g\text{Best}_d$ is the dth variable of the global best position found by the overall particles so far; c_1 and c_2 are acceleration parameters which are commonly set to 2.0; r_{1d} and r_{2d} are two random numbers drawn from a uniform distribution over [0, 1]; and w denotes the inertia weight, which is used to set up the balance between the abilities of global and local search features of PSO. The inertia weight parameter is widely adopted by major PSO variants [31].

The behavior of the particle is specified by its velocity and position update realized according to (1) and (2) [39, 41]. The first inertia weight component of (1) models the tendency of the particle to continue in the same direction as before. The second component of (1) is referred to as the particle's "memory," "self-knowledge," "nostalgia," or "remembrance" [39, 41]. It reflects the self-learning behavior of the particle. The third component in (1) is referred to as "cooperation," "social knowledge," "group knowledge," or "shared information" [39, 41]. It reflects the social learning behavior of the particle. Equation (2) indicates that the position of the particle in the solution space will be changed in terms of its current position and next velocity.

After each update, we check the position and velocity of each particle to guarantee them being within a predefined certain range. In our study, if the position and velocity exceed the range, they are modified as follows:

$$v'_{id} \longleftarrow \min\left(v_d^{\max}, \max\left(v_d^{\min}, v'_{id}\right)\right),$$
$$x'_{id} \longleftarrow \overline{p\text{Best}_{id}}, \tag{3}$$

where $v_d^{\max}$ and $v_d^{\min}$ are maximum and minimum value of dth variable of the velocities, respectively. $\overline{p\text{Best}_{id}}$ is the mean value of dth variable of the personal historical best positions of all particles.

2.2. Major PSO Variants. "Standard" PSO exhibits some deficiencies, including suffering from being premature and inefficient in solving complex multimodal optimization problems. One way to strengthen the capability of PSO is to dynamically adapt its parameters when running the particles' evolutionary process. The inertia weight parameter w is set to linearly decrease over iterations [31, 42]. In addition, a fuzzy adaptive mechanism was used to tune the value of w [9]. Kennedy and Eberhart recommended that the proper value for the acceleration parameters c_1 and c_2 could be fixed and set to 2.0. These values were adopted in many works. In comparison, Suganthan [13] suggested that the usage of ad hoc selected values for c_1 and c_2 rather than the fixed value for different problems could result in better performance. Ratnaweera et al. [8] presented a PSO variant with linearly time-varying acceleration coefficients (HPSO-TVAC). Zhan et al. [17] proposed an adaptive PSO, which enables the automatic control of inertia weight, acceleration coefficients, and other algorithmic parameters at run time according to four evolutionary states, that is, exploration, exploitation, convergence, and jumping out state. Ismail and Engelbrecht [10] controlled the parameters of PSO by embedding them in the position vector of particles, which enhanced the performance of comprehensive learning PSO (CLPSO) [35].

Besides parameter adaptation, topological structures of the particle swarm were also extensively studied. For example, Kennedy [12, 16] suggested that a small neighborhood might be more suitable to complicated multimodal problems while a larger neighborhood might be more effective for relatively simple unimodal problems. In [16], Kennedy and Mendes evaluated some typical topologies including global best topology, ring topology, wheel topology, pyramid topology, and Von Neumann topology. They suggested that the Von Neumann topology configuration may perform better compared to others. However, the selection of an appropriate neighborhood structure is generally problem oriented. Being aware of the noticeable effect of neighborhood structures, the neighborhood structure dynamic adaptation mechanisms were also investigated by some researchers [13, 43]. Mendes et al. [33] presented a fully informed particle swarm (FIPS) in which each individual learns the experience of all its neighbors rather than just the best one and itself.

Another natural evolution of the Particle Swarm Optimization can be achieved by incorporating operators or techniques that are effectively used in other evolutionary algorithms [39]. Angeline [20] developed a hybrid PSO by introducing the selection operator coming from Genetic Algorithm. A hybrid PSO based on genetic programming was presented by Poli et al. [23]. In [44], Juang integrated GA with PSO for designing artificial neural network. Other operators and techniques, such as crossover [45], mutation [46], local search [15], and differential evolution [47, 48], were adopted in PSO as well.

An intelligent integration of different learning strategy in to the swarm evolutionary process is a promising direction for designing efficient PSO variants. Usually one prepares a collection of learning strategies, which possess different capabilities, such as exploitation, exploration, and jumping out from local optimum, and then, through a sophisticated adaptation mechanism, enables each particle to automatically choose learning strategies to determine the next move. Many state-of-the-art PSO variants have been developed following this development strategy. Liang et al. [35] proposed a comprehensive learning particle swarm optimizer (CLPSO), which uses a novel learning strategy whereby all other particles' historical best information is used to update a particle's velocity. Wang et al. [24] proposed a self-adaptive learning based PSO (SLPSO). SLPSO adopts four adaptive learning mechanisms, which are automatically chosen by particles based on each strategy's past performance. In [4], Zhan et al. proposed an orthogonal learning (OL) strategy for PSO to discover more useful information that lies in two particles' experiences via orthogonal experimental design. Experimental results demonstrated that OLPSO significantly improves the performance of PSO, offering faster convergence, higher solution quality, and stronger robustness. Hu et al. [18] proposed a PSO variant by intelligently combining a nonuniform mutation-based method and an adaptive subgradient method. A Cauchy mutation operator was further utilized to prevent premature convergence. Wang et al. [49] presented

an enhanced PSO variant called GOBL, which employed generalized opposition-based learning (GOBL) and Cauchy mutation to overcome the deficiency of premature. Li et al. [19] presented a self-learning particle swarm optimizer, in which each particle has four strategies to cope with different situations in the search space. An adaptive cooperation mechanism was implemented at the individual level, which enables a particle to choose the rational strategy according to its own local fitness landscape.

3. The Inner Variable Learning Strategy

In this section, we introduce the knowledge employed in the inner variable learning (IVL) strategy and discuss the detailed implementation of the IVL strategy.

3.1. Knowledge Employed in the Inner Variable Learning Strategy. As mentioned before, adaptive learning is an important concept in designing evolutionary algorithms. In the basic PSO, each particle flies through the search space aiming to obtain a satisfactory solution, with its velocity and position being dynamically updated referring to its flying experience and its companions' experience. Two typical learning strategies are included in basic PSO: the first one is the self-learning strategy, which enables each particle to consider its past velocity and personal local best position when determining the next search direction and speed; the second one is the companion learning strategy, by which each particle takes into account the flying experience of its companions (such as learning from the global best position or all its neighbors' positions) in its space search process. It can be found from the review in Section 2 that current learning strategies (or cooperation and interaction) of PSO mainly happen at the swarm or particle level. However, the learning strategy at the variable level is rarely studied. We think that variable level based learning mechanisms would be more effective, since different variables of a particle are evolved independently. In addition, it is also meaningful to extract useful knowledge from the optimization problem to provide more exact and effective guidance for the search behavior of particles.

We find that, in many optimization functions, different variables come in the same form; that is, they are symmetric. Such functions usually combine the variables by using the operators of summation and product ("$\sum$" and "$\prod$"). For example, with regard to the Rosenbrock function: $f(\mathbf{x}) = \sum_{i=1}^{D}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$, this function is multimodal and nonseparable and exhibits a very narrow valley moving from local optimum to global optimum [50]. Note that different variables in the Rosenbrock function are symmetric, since we can exchange the positions of any two variables x_i and x_j without affecting the function. Then, in the optimal solution, any two variables x_i and x_j are supposed to satisfy the relationship $x_i = x_j$. More generally, let us consider an optimization function $f(\mathbf{ax} - \mathbf{b})$, where $\mathbf{a} = [a_1, a_2, \ldots, a_D]$, $\mathbf{x} = [x_1, x_2, \ldots, x_D]$, and $\mathbf{b} = [b_1, b_2, \ldots, b_D]$. Let us formally define the concept of variable symmetry.

Variable Symmetry. two variables are symmetric if they can exchange their positions in the function through some linear transformation without affecting this function.

For instance, with regard to the two variables x_i and x_j in a given function, if we exchange these two variables by letting $x_i = (a_j x_j + b_j - b_i)/a_i$ and $x_j = (a_i x_i + b_i - b_j)/a_j$ without changing the function, then we say variables x_i and x_j are symmetric. If x_i and x_j are symmetric, then, in the optimal solution, there should exist the relationship $a_i x_i + b_i = a_j x_j + b_j$. As an example, let us take the Shift Rastrigin function. In Rastrigin $f_2(\mathbf{x}) = \sum_{i=1}^{n}(y_i^2 - 10\cos(2\pi y_i) + 10)$, $y_i = x_i - o_i$, $\mathbf{o}$ is a shift vector. Consider the original Shift Rastrigin function with two variables:

$$\begin{aligned}&\left((x_1 - o_1)^2 - 10\cos\left(2\pi\left(x_1 - o_1\right)\right) + 10\right)\\ &\quad + \left((x_2 - o_2)^2 - 10\cos\left(2\pi\left(x_2 - o_2\right)\right) + 10\right).\end{aligned} \tag{4}$$

Then we let $x_1 = x_2 - o_2 + o_1$ and $x_2 = x_1 - o_1 + o_2$ and substitute them into the above form:

$$\begin{aligned}&\left((x_2 - o_2 + o_1 - o_1)^2 - 10\cos\left(2\pi\left(x_2 - o_2 + o_1 - o_1\right)\right) + 10\right)\\ &\quad + \left((x_1 - o_1 + o_2 - o_2)^2\right.\\ &\qquad \left. -10\cos\left(2\pi\left(x_1 - o_1 + o_2 - o_2\right)\right) + 10\right)\end{aligned} \tag{5}$$

which gives rise to the expression

$$\begin{aligned}&\left((x_2 - o_2)^2 - 10\cos\left(2\pi\left(x_2 - o_2\right)\right) + 10\right)\\ &\quad + \left((x_1 - o_1)^2 - 10\cos\left(2\pi\left(x_1 - o_1\right)\right) + 10\right).\end{aligned} \tag{6}$$

It becomes clear that (6) and (4) are equal. The same situation happens for any other two variables. Therefore, the Shift Rastrigin function is completely symmetric. Different variables in the optimal solution of the Shift Rastrigin function have to satisfy $x_i - o_i = x_j - o_j$.

It should be noted that the relation of variable symmetry is reflexive and transitive. We can determine the property of variable symmetry of the optimized function by exchanging positions of any two variables and checking the properties of reflexivity and transitivity. Sometimes, intuitive hints are also helpful.

Having noted the knowledge that symmetric variables of the optimal solution of a function should satisfy a certain quantitative relation, we can develop an inner variable learning (IVL) strategy in which different variables in the same function can realize learning from each other during the problem solving process. The idea of this learning strategy is simple and straightforward. Namely, in the course of learning, we check the variables of a particle and determine which variable's value is the best exemplar of other variables to optimize the function to the highest extent.

3.2. Implementation of the Inner Variable Learning Strategy. The previous learning strategies, such as the self-learning strategy and companion learning strategy, enable particles to

learn their past flying experience or their companions' past flying experience, which are at the swarm or particle level. In comparison, the new learning strategy to be presented here is realized at the variable level and thus referred to as inner variable learning (IVL) strategy. The IVL strategy enables a particle to inspect the relation among its variables of the position, determine the exemplar variable for other variables, and then make each variable learn from the exemplar variable by modifying the values of other variables in terms of their quantitative relation with the exemplar variable and the value of the exemplar variable. This strategy will lead particles to fly to a better position quickly. Note that this learning strategy has originated from the knowledge of variable symmetry of optimized functions, such that it can be applied to any function involving symmetric variables.

If we execute the IVL strategy on each particle at every generation of PSO, it could be a little time consuming since every time we need to evaluate the effectiveness of each variable and select out an exemplar variable. In addition, performing the IVL strategy too frequently may cause PSO to suffer from premature convergence and make it get trapped in a local optimum at the early stage. That is because once a particle executes the IVL strategy, all its variables will be directly modified according to the value of the exemplar variable. In this study, the particle executes the IVL strategy immediately after it visits the personal best position or jumps out from a local optimum. This is because under these two occasions, the particle may have potential high-quality exemplar variable.

At each evolutionary generation, once the particle i determines its personal best position or executes the trap jumping operation, it will execute the inner learning strategy accordingly. Assume that the current personal best position and the corresponding velocity of particle i are $\mathbf{x_i} = [x_{i1}, x_{i2}, \ldots, x_{iD}]$ and $\mathbf{v_i} = [v_{i1}, v_{i2}, \ldots, v_{iD}]$. We test the effectiveness of every variable x_{id} in the optimization function and take the best variable as the exemplar variable. That is to say, the exemplar variable is determined by first trying to let every variable be the exemplar variable and then ascertaining the best one as the ultimate exemplar variable. To obtain the effectiveness of variable x_{id}, we just let x_{id} be the temporal exemplar variable, modify the value of any other variable x_{ik} according to $x_{ik} = (a_d x_{id} + b_d - b_k)/a_k$, and then calculate the function fitness. The temporary exemplar variable resulting in the best function fitness will be the ultimate exemplar variable. Assume that the ultimate exemplar variable is denoted by d. The procedure of the IVL strategy of particle i is described in Algorithm 1. Note that when a particle executes the IVL strategy once, it performs D evaluations of the fitness function.

4. The Trap Detection and Jumping out Techniques

The PSO algorithm is easy to implement and has been empirically shown to perform well on numerous optimization problems. However, it may easily get trapped in a local optimum when solving complex multimodal problems [35], such that effective mechanisms for particle detecting and jumping out of trap state become necessary.

Many researchers noted that it is helpful to improve the performance of PSO by intelligently tuning the particle's behaviors according to the current evolutionary states, which is usually evaluated by the statistic information of the swarm's distribution. For instance, Zhan et al. [17] determined the evolutionary states (i.e., exploration, exploitation, convergence, and jumping out) with the statistic of the position distribution information of the population, which was used for guiding the automatic parameter adjustment. The distribution information was obtained by calculating the mean distance of each particle to all the other particles. Chen et al. [51] also used the distance between particles to evaluate the diversity of PSO in the evolutionary process. They incorporated diversity into the objective function to optimize the optimization problem as well as guarantee the diversity of the overall swarm. It can be seen that the above methods are established at the swarm level, which means that the authors checked the diversity (or distribution information) of the overall swarm each time and adjusted the behaviors of particles accordingly. This may be not good for the adaptation and flexibility of a single particle, though. Although sometimes the diversity of the whole swarm is satisfactory according to certain criteria, some of the particles may actually have been trapped in optima.

To enable a particle to react to its solution space search situation more efficiently, we will check the diversity of particle i when $g\text{Best}_i$ has not been improved continuously for M generations. In addition, let m_i denote the number of stagnation generation of particle i. At each generation, if the $g\text{Best}_i$ has not improved, m_i is increased by 1; otherwise, m_i will be set to 1. Two criteria are considered to determine whether the particle is trapped in a local optimum.

Now let us take particle i as an example. The first criterion concerns the difference between the function fitness of the previous position $\mathbf{x}_i''$ and that of the current position $\mathbf{x_i}$ of particle i. Let δ denote a threshold value of the difference of function fitness. In our study, we set $\delta = \alpha \cdot t/\text{maxEF}$, where maxEF denotes the maximum number of function fitness evaluations, t denotes the current consumed number of function fitness evaluations, and α is scale coefficient. δ changes adaptively with the evolution of particles. If the function fitness difference is lower than δ, the particle may be considered to be trapped in a local optimum:

$$\left| f\left(\mathbf{x}_i''\right) - f\left(\mathbf{x}_i\right) \right| < \delta. \tag{7}$$

The second criterion is as follows: if the distance between two consecutive positions of particle i is smaller than a predefined threshold value ξ, particle i may have been trapped:

$$\left\| \mathbf{x}_i'' - \mathbf{x}_i \right\| = \frac{\sqrt{\sum_d^D \left(x_{id}'' - x_{id}\right)^2}}{D < \xi}. \tag{8}$$

We set $\xi = \sqrt{D \cdot \beta^2}$, where D is the number of variables and β is the scale coefficient.

/∗particle i has reached its personal best position∗/
$fitness = f(\mathbf{a}\mathbf{x}_i + \mathbf{b})$;
$d = 1$;
For $j = 1: D$
/∗every jth variable as the temporal exemplar variable∗/
  For $k = 1: D$

$$\widehat{x}_{ik} = \frac{(a_j x_{ij} + b_j - b_k)}{a_k};$$

  End for
  If $f(\mathbf{a}\widehat{\mathbf{x}}_i + \mathbf{b}) < fitness$
    $fitness = f(\mathbf{a}\widehat{\mathbf{x}}_\mathbf{i} + \mathbf{b})$;
    $d = j$; /∗d records the best exemplar variable up to now∗/
  End if
End for
/∗all other variables are modified in terms of the exemplar variable marked by d∗/
For $j = 1: D$

$$x_{ij} = \frac{(a_d x_{id} + b_d - b_j)}{a_j};$$

End for
If $f(\mathbf{a}\mathbf{x}_i + \mathbf{b}) < f(p\text{Best}_i)$
    $p\text{Best}_i = \mathbf{x}_\mathbf{i}$;
End if
If $f(\mathbf{a}\mathbf{x}_i + \mathbf{b}) < f(g\text{Best})$
    $g\text{Best} = \mathbf{x}_i$;
End if

Algorithm 1: InnerVariableLearning(i).

However, none of above two criterions can judge the trap state independently. That is because, on the one hand, as for the first criterion, a particle may have very close function fitness at two distant positions (meaning the particle is not trapped). On the other hand, the optimization function may be very sensitive to the landscape, so small deviation of the position of a particle may result in significant difference (similarly, meaning the particle is not trapped) of the function fitness. Therefore, the two criterions should be taken into account simultaneously. And if both of the above criterions are met, particle i is safely considered to be trapped in a local optimum.

Once we detect that a particle is trapped, a mutation operator will be employed to help particles to escape from the local optimum. Mutation is an indispensable operator in Genetic Algorithm. Due to the effectiveness of mutation operator in enhancing the diversity of population-based algorithms, it is also popularly adopted in many PSO variants. Generally, Cauchy mutation [18, 51] and Gaussian mutation [17, 49] methods are mostly used. Andrews [46] utilized a PSO algorithm incorporating different mutation operators to cope with both mathematical and constrained optimization problems. His results showed that the addition of a mutation operator to PSO could enhance optimization performance and insight was gained into how to design mutation operators dependent on the nature of the problem being optimized. The Gaussian mutation operator is utilized in the discussed PSO variant:

$$x'_{id} = x_{id} + \omega \cdot \left(x_{\max,d} - x_{\min,d}\right) \cdot \lambda \cdot \text{Gaussian}\,(0, 1)\,, \quad (9)$$

where $x_{\max,d}$ and $x_{\min,d}$ stand for the upper and low bound of the dth variable of the optimization problem and ω and λ are coefficients controlling the mutation scale:

$$\lambda = 1 - \frac{\sigma \cdot t}{\text{maxEF}}. \quad (10)$$

Like the inertia weight parameter, mutation scale parameter λ linearly decreases with the evolutionary process, that is, starts declining from 1.0 to $(1 - \sigma)$ gradually. σ $(0 < \sigma < 1)$ reflects the decreased speed. The linear decrease of the mutation scale parameter enables the PSO to exhibit higher exploration capability at the early stage of the evolution and strong exploitation ability at the later evolutionary stage.

In addition, if $g\text{Best}_i$ has not been improved for each N successive generation, particle i will perform the mutation operator. The PSO variant integrated with the IVL strategy, trap detection and jumping out strategy, and the basic PSO is shown as Algorithm 2.

5. Experimental Tests

5.1. Experimental Setting. In order to evaluate the performance of PSO-IVL, we compare it with some other state-of-the-art PSO alternatives. The parameters of the algorithms selected for comparison are summarized in Table 1. For comparison, the experimental settings of benchmark PSO variants are similar to that of [24]; that is, the population size of particles is 50 and the number of variables (dimensions) of each test function is set to 30, which is a typical setting encountered in the literature. Choosing proper parameters

Initialize the position $\mathbf{x}_i$ and velocity $\mathbf{v}_i$ for each particle i;
Let $g\text{Best}_i \leftarrow \mathbf{x_i}$ and calculate the gBest;
Set $t = 0$;
Initialize parameters ps, δ, ξ, maxEF, M and N
While ($t <$ maxEF)
  For i =1: ps
    If $m_i\%M == 0$ /∗determine whether particle i gets trapped∗/
      If $\left|f(\mathbf{ax}_i'' + \mathbf{b}) - f(\mathbf{ax}_i + \mathbf{b})\right| < \delta$ && $||\mathbf{x}_i'' - \mathbf{x}_i|| = \sqrt{\sum_d^D (x_{id}'' - x_{id})^2}/ps < \xi$
        $\lambda = 1 - 0.9 \cdot t/\text{maxEF}$
        $x_{id} = x_{id} + \omega \cdot (x_{\max,d} - x_{\min,d}) \cdot \lambda \cdot \text{Gaussian}(0, 1)$
        Execute the strategy of ***InnerVariableLearning***(i);
        $t = t + D$; /∗increase the number of fitness evaluations∗/
      End if
    End if
      If $m_i\%N == 0$
        $x_{id} = x_{id} + \omega \cdot (x_{\max,d} - x_{\min,d}) \cdot \lambda \cdot \text{Gaussian}(0, 1)$
        Execute the strategy of ***InnerVariableLearning***(i);
        $t = t + D$;
      End if
      /∗Update the position and velocity of the ith particle∗/
      $v_{id} = w \times v_{id} + c_1 \times r_{1d} \times (pBest_{id} - x_{id}) + c_2 \times r_{2d} \times (gBest_d - x_{id})$;
      $x_{id} = x_{id} + v_{id}$
      $t = t + 1$;
      If $f(\mathbf{ax}_i + \mathbf{b}) < f(p\text{Best}_i)$
        $p\text{Best}_i \leftarrow \mathbf{x}_i$;
        Execute the strategy of ***InnerVariableLearning***(i);
        $t = t + D$;
        $m_i = 1$
      Else
        $m_i = m_i + 1$;
      End if
      If $f(\mathbf{ax}_i + \mathbf{b}) < f(g\text{Best})$
        $g\text{Best} = \mathbf{x}_i$;
      End if
  End for
  If gBest is the optimal solution
    Break;
  End if
  $w = 0.9 - 0.5 \cdot t/\text{maxEva}$;
End while

Algorithm 2: Procedure of PSO-IVL.

Table 1: PSO variants used in comparative studies.

PSO variants	Parameters setting
PSO-w: PSO with inertia weight [31]	$w = 0.9 - \dfrac{0.5 \cdot g}{\text{maxGen}}$, $f_1 = f_2 = 1.49$
PSO-cf: PSO with constriction factor [32]	$w = 0.729$, $c_1 = c_2 = 1.49445$
PSO-cf-local: local version of PSO with constriction factor [16]	$w = 0.729$, $c_1 = c_2 = 1.49445$
FIPS-PSO: fully informed PSO [33]	$w = 0.729$, $c_1 = c_2 = 2.0$
CPSO-H: cooperative based PSO [34]	$w = 0.9 - \dfrac{0.5 \cdot g}{\text{maxGen}}$, $c_1 = c_2 = 2.0$
CLPSO: comprehensive learning PSO [35]	$w = 0.9 - \dfrac{0.5 \cdot g}{\text{maxGen}}$, $c_1 = c_2 = 1.49445$
SLPSO: self-adaptive learning based Particle Swarm Optimization [24]	$proSTR_i = 0.25$, $Gs = 10$, $w_i = \log(ps - i + 1)/(\log(1) + \cdots + \log(ps))$

for an evolutionary algorithm is always time consuming since parameters are always related. According to analysis of extensive experimental work, the parameters of the PSO-IVL algorithm are specified as follows: $ps = 20$, maxEF $= 300,000$, $c_1 = c_2 = 2.0$, $\alpha = 0.1$, $\delta = 0.1 \cdot t/\text{maxEF}$, $\beta = 0.1$, $\xi = \sqrt{0.01D}$, $\omega = 0.05$, $\sigma = 0.9$, $\lambda = 1 - (0.9 \cdot t/\text{maxEF})$, $M = 50$, and $N = 200$.

To realize a comprehensive analysis of the PSO-IVL and other PSO variants listed above, we conducted a series of experiments by employing 18 classical numerical optimization problems with different characteristics, including unimodality, multimodality, rotation, ill-conditionality, misscale, and noise. The optimization functions used in the experiments are listed below; $\mathbf{M}$ is the orthogonal matrix and $\mathbf{o}$ is the shifted vector:

Sphere: $f_1(x) = \sum_{i=1}^{n} y_i^2$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Rastrigin: $f_2(\mathbf{x}) = \sum_{i=1}^{n} (y_i^2 - 10\cos(2\pi y_i) + 10)$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Rosenbrock: $f_3(\mathbf{x}) = \sum_{i=1}^{n} (100(y_i^2 - y_{i+1})^2 + (y_i - 1)^2)$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Griewank: $f_4(\mathbf{z}) = (1/4000) \sum_{i=1}^{n} y_i^2 + 1 - \prod_{i=1}^{n} \cos(y_i/\sqrt{i})$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Ackley: $f_5(x) = -20 \cdot \exp(-0.2\sqrt{(1/n) \sum_{i=1}^{n} y_i^2}) + 20 - \exp((1/n) \sum_{i=1}^{n} \cos(2\pi y_i)) + e$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Schwefel 1.2: $f_6(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} y_j^2)$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Scaled Rosenbrock 100: $f_7(x) = \sum_{i=1}^{n} (100((a_i y_i)^2 - (a_{i+1} y_{i+1}))^2 + (a_i y_i - 1)^2)$, $a_i = 100^{(i-1)/(n-1)}$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Scaled Rastrigin 10: $f_8(\mathbf{x}) = \sum_{i=1}^{n} ((a_i y_i)^2 - 10\cos(2\pi (a_i y_i)) + 10)$, $a_i = 10^{(i-1)/(n-1)}$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Noise Schwefel 1.2: $f_9(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} y_j^2) \cdot (1 + 0.4|N(0,1)|)$, $\mathbf{y} = \mathbf{x} - \mathbf{o}$,

Rotated Sphere: $f_{10}(x) = \sum_{i=1}^{n} z_i^2$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Rotated Schwefel 2.21: $f_{11}(x) = \max |z_i|$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Rotated Ellipse: $f_{12}(x) = \sum_{i=1}^{n} (20^{(i-1)/(n-1)} z_i)^2$, $\mathbf{z} = \mathbf{M} \cdot \mathbf{x}$,

Rotated Rosenbrock: $f_{13}(\mathbf{x}) = \sum_{i=1}^{n} (100(z_i^2 - x_{i+1})^2 + (z_i - 1)^2)$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Rotated Ackley: $f_{14}(x) = -20 \cdot \exp(-0.2\sqrt{(1/n) \sum_{i=1}^{n} z_i^2}) + 20 - \exp((1/n) \sum_{i=1}^{n} \cos(2\pi z_i)) + e$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Rotated Griewank: $f_{15}(\mathbf{z}) = (1/4000) \sum_{i=1}^{n} z_i^2 + 1 - \prod_{i=1}^{n} \cos(z_i/\sqrt{i})$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Rotated Rastrigin: $f_{16}(\mathbf{x}) = \sum_{i=1}^{n} (z_i^2 - 10\cos(2\pi z_i) + 10)$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Noise Rotated Schwefel 1.2: $f_{17}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} z_j^2) \cdot (1 + 0.4|N(0,1)|)$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$,

Noise Rotated Quadric: $f_{18}(x) = \sum_{i=1}^{n} i z_i^4 + \text{random}[0,1)$, $\mathbf{z} = \mathbf{M} \cdot (\mathbf{x} - \mathbf{o})$.

5.2. Comparative Analysis. The simulation results for each optimized function produced by PSO-w, PSO-cf, PSO-cf-local, FIPS-PSO, CPSO-H, CLPSO, and SLPSO are reported from [24]. Each optimization function is run by each PSO variant 30 times. The computational results are listed in Table 2 including the average value of the results along with their standard deviation. Suc denotes the number of successful runs. According to [24], a run is considered to be successful (i.e., has obtained a satisfactory solution) if a solution is obtained whose fitness value is not worse than $(fit(x^*) + (1.0E - 5))$, where x^* is the theoretical global optimal solution. FEs denotes the average number of function evaluations required to find the satisfactory solution when all 30 runs are successful.

From the computational results given in Table 2, we can conclude that PSO-IVL produced the best result for every test function. However, for Sphere function $f_1(x)$, Ackley function $f_5(x)$, Rotated Sphere function $f_{10}(x)$, and Rotated Ackley function $f_{14}(x)$, although PSO-IVL can find the optimal solution, its efficiency is not the highest. Moreover, PSO-IVL can find the optimal solution for all optimization functions only except Scaled Rosenbrock 100 $f_7(x)$, Rotated Rosenbrock $f_{13}(\mathbf{x})$, and Noise Quadric $f_{18}(x)$. Especially for some noisy and rotated functions, such as Noise Schwefel 1.2 $f_9(x)$, Rotated Ellipse $f_{12}(x)$, Rotated Rastrigin $f_{16}(\mathbf{x})$, and Noisy Rotated Schwefel 1.2 $f_{17}(x)$, other peer PSO variants cannot effectively acquire satisfactory solutions; however, PSO-IVL is successful in these cases. As a result, based on the reported results, Table 2, we conclude that the performance of PSO-IVL reported on the test functions is fairly competitive compared to other PSO variants.

5.3. Convergence Analysis of PSO-IVL. To provide an intuitive illustration of the optimization behavior of PSO-IVL, in Figure 1, we display the evolutionary process of a particle and the global-best-so-far solution when PSO-IVL is utilized to solve each optimization function. It should be noted that in the basic PSO and many typical PSO variants, for a particle, the number of generations and the number of fitness evaluations are usually equal. However, the situation is different in our study. At any generation, if the particle does not perform the IVL strategy, a single fitness evaluation is required; thus in this case one generation is corresponding to one fitness evaluation. In comparison, if the particle executes the IVL strategy at a given generation, then there are D (D is the number of variables associated with the functions considered) numbers of fitness evaluation operation to complete. As a result, in this situation, one generation is related to D of fitness evaluations. The evolutionary process of other PSO alternatives can be found in [24].

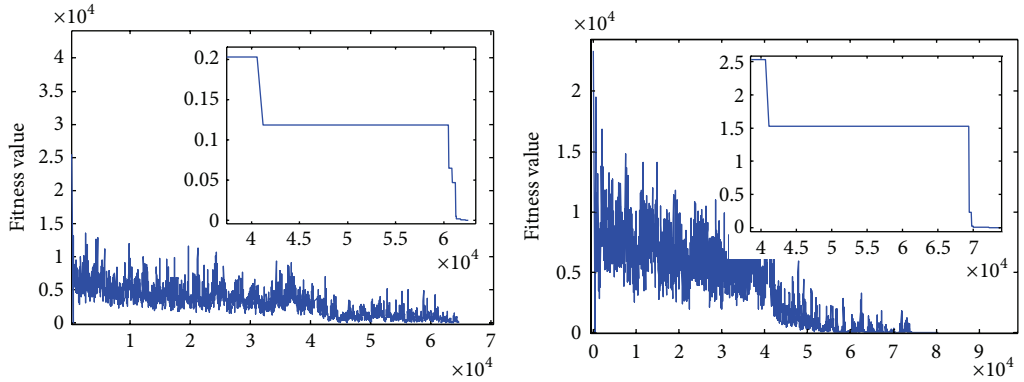
The larger figure shows how the fitness of the position visited by a particle in PSO-IVL changes with the increase of the number of function fitness evaluations while the smaller figure visualizes how the fitness of the global-best-so-far solution evolves. Since the global-best-so-far solution converges to a good value very fast, it would be unclear to see

Table 2: Optimization results obtained for the test functions; the best results are shown in boldface.

Functions	Mean	StdDev	Suc	FEs	Mean	StdDev	Suc	FEs	Mean	StdDev	Suc	FEs
Algorithm		Sphere				Rastrigin				Rosenbrock		
PSO-w	$0.00E+00$	$0.00E+00$	30	193,017	$1.75E+01$	$5.97E+00$	0		$6.46E+01$	$8.01E+01$	0	
PSO-cf	**$0.00E+00$**	**$0.00E+00$**	**30**	**20,333**	$6.45E+01$	$2.05E+01$	0		$8.88E+00$	$1.46E+01$	0	
PSO-cf-local	$0.00E+00$	$0.00E+00$	30	40,295	$4.10E+01$	$1.13E+01$	0		$2.98E+01$	$4.07E+01$	0	
FIPS-PSO	$0.00E+00$	$0.00E+00$	30	146,179	$6.39E+01$	$1.12E+01$	0		$2.52E+01$	$9.08E-01$	0	
CPSO-H	$0.00E+00$	$1.49E-08$	30	149,044	$3.32E-02$	$1.82E-01$	29		$2.77E+01$	$2.86E+01$	0	
CLPSO	$0.00E+00$	$0.00E+00$	30	122,161	$0.00E+00$	$0.00E+00$	30	195,815	$4.95E+00$	$3.79E+00$	0	
SLPSO	$0.00E+00$	$0.00E+00$	30	43,980	$0.00E+00$	$0.00E+00$	30	196,749	$2.66E-01$	$1.01E+00$	8	
PSO-IVL	$0.00E+00$	$0.00E+00$	30	81,950	**$0.00E+00$**	**$0.00E+00$**	**30**	**99,430**	**$0.00E+00$**	**$0.00E+00$**	**30**	**136,250**
Algorithms		Griewank				Schwefel 1.2				Ackley		
PSO-w	$8.73E-02$	$1.18E-01$	2		$0.00E+00$	$1.11E+00$	0		$2.18E+05$	$4.83E-14$	30	211,209
PSO-cf	$1.71E-02$	$1.78E-02$	8		$1.29E-12$	$3.88E-11$	30	151,095	$8.51E-01$	$1.01E+00$	15	
PSO-cf-local	$5.34E-03$	$7.46E-03$	17		$1.53E-04$	$1.68E-04$	1		$0.00E+00$	$0.00E+00$	30	56,976
FIPS-PSO	$2.72E-07$	$1.18E-06$	30	183,581	$2.08E+02$	$8.98E+01$	0		$1.39E-08$	$2.98E-09$	30	
CPSO-H	$1.20E-01$	$2.18E-01$	4		$2.79E+03$	$5.98E+03$			$2.44E-05$	$1.35E-05$	1	
CLPSO	$0.00E+00$	$0.00E+00$	30	151,708	$1.16E+03$	$2.44E+02$	0		$7.77E-13$	$1.49E-13$	30	166,425
SLPSO	$1.81E-03$	$4.79E-03$	26		$7.69E-13$	$4.27E-13$	30	149,872	**$0.00E+00$**	**$0.00E+00$**	**30**	**51,585**
PSO-IVL	**$0.00E+00$**	**$0.00E+00$**	**30**	**81,960**	**$0.00E+00$**	**$0.00E+00$**	**30**	**92,600**	$0.00E+00$	$0.00E+00$	30	83,330
Algorithms		Scaled Rosenbrock 100				Scaled Rastrigin 10				Noise Schwefel 1.2		
PSO-w	$2.18E+05$	$8.24E+05$	0		$2.18E+05$	$6.59E+00$	0		$4.68E+02$	$3.14E+02$	0	
PSO-cf	$2.57E+04$	$4.91E+04$	0		$2.57E+04$	$2.85E+01$	0		$1.99E+02$	$2.89E+02$	0	
PSO-cf-local	$9.11E+04$	$3.72E+05$	0		$2.57E+04$	$1.35E+01$	0		$5.71E+02$	$4.59E+02$	0	
FIPS-PSO	$7.37E+04$	$3.14E+05$	0		$7.37E+04$	$9.23E+00$	0		$1.52E+03$	$5.44E+02$	0	
CPSO-H	$3.71E+06$	$4.38E+06$	0		$1.23E+07$	$1.86E-07$	0		$2.44E+04$	$8.49E+03$	0	
CLPSO	$1.09E+03$	$3.45E+03$	0		$0.00E+00$	$0.00E+00$	30	226,863	$7.25E+03$	$1.37E+03$	0	
SLPSO	$7.88E+02$	$2.56E+03$	0		$0.00E+00$	$0.00E+00$	30	234,253	$2.32E-02$	$8.92E-02$	0	
PSO-IVL	**$2.32E+01$**	**$2.54E-03$**	**0**		**$0.00E+00$**	**$0.00E+00$**	**30**	**83,660**	**$0.00E+00$**	**$0.00E+00$**	**30**	**97,530**
Algorithms		Rotated Sphere				Rotated Schwefel 2.21				Rotated Ellipse		
PSO-w	$0.00E+00$	$4.56E-14$	30	201,639	$2.50E-01$	$3.03E-01$	0		$1.15E+02$	$1.49E+02$	0	
PSO-cf	**$0.00E+00$**	**$0.00E+00$**	**30**	**24,010**	$4.11E-02$	$1.40E-01$	17		$3.46E-03$	$1.13E-02$	0	
PSO-cf-local	$0.00E+00$	$0.00E+00$	30	47,289	$7.98E-02$	$2.03E-01$	14		$7.66E-01$	$1.09E+00$	0	
FIPS-PSO	$7.54E-13$	$3.26E-13$	30		$1.36E-04$	$4.89E-05$	0		$1.51E+03$	$7.14E+02$	0	
CPSO-H	$8.10E-08$	$1.02E-07$	30		$5.43E+01$	$7.52+00$	0		$7.63E+03$	$6.69E+03$	0	
CLPSO	$4.21E-10$	$6.18E-10$	30	190,125	$9.71E-01$	$2.38E-01$	0		$4.07E+03$	$9.37E+02$	0	
SLPSO	$0.00E+00$	$0.00E+00$	30	49,396	$4.51E-11$	$1.16E-10$	30	81,481	$2.22E-12$	$7.06E-13$	30	146,787
PSO-IVL	$0.00E+00$	$0.00E+00$	30	84,750	**$0.00E+00$**	**$0.00E+00$**	**30**	**129,880**	**$0.00E+00$**	**$0.00E+00$**	**30**	**97800**
Algorithms		Rotated Rosenbrock				Rotated Ackley				Rotated Griewank		
PSO-w	$4.62E+05$	$7.88E+05$	0		$2.34E+00$	$7.59E-01$	1		$2.07E-01$	$3.84E-01$	0	
PSO-cf	$1.18E+03$	$2.98E+03$	0		$1.95E+00$	$9.55E-01$	4		$9.85E-03$	$7.99E-03$	8	
PSO-cf-local	$6.16E+02$	$1.93E+03$	0		$2.40E-01$	$4.98E-01$	24		$1.04E-02$	$1.03E-02$	9	
FIPS-PSO	$2.89E+01$	$4.15E+00$	0		$2.24E-08$	$5.60E-09$	30	213,274	$1.14E-03$	$3.00E-03$	8	
CPSO-H	$1.62E+03$	$3.78E+03$	0		$1.76E+01$	$3.96E+00$	0		$1.66E+00$	$2.10E-01$	0	
CLPSO	$2.56E+02$	$3.04E+02$	0		$1.16E-02$	$9.10E-03$	0		$3.36E-02$	$2.00E-02$	0	
SLPSO	$1.20E+02$	$3.81E+02$	0		**$0.00E+00$**	**$0.00E+00$**	**30**	**52,575**	$5.34E-03$	$6.82E-03$	17	
PSO-IVL	**$2.38E+01$**	**$2.80E-03$**	**0**		$0.00E+00$	$0.00E+00$	30	95,230	**$0.00E+00$**	**$0.00E+00$**	**30**	**84910**

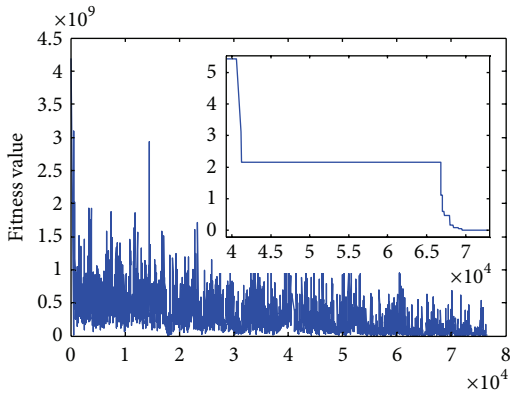
Table 2: Continued.

Functions Algorithm	Mean	StdDev Sphere	Suc	FEs	Mean	StdDev Rastrigin	Suc	FEs	Mean	StdDev Rosenbrock	Suc	FEs
Algorithms		Rotated Rastrigin				Noise Rotated Schwefel1.2				Noise Quadric		
PSO-w	$8.29E+01$	$5.00E+01$	0		$4.88E+02$	$3.51E+02$	0		$1.64E-02$	$6.28E-03$	0	
PSO-cf	$1.09E+02$	$3.63E+01$	0		$2.39E+02$	$2.97E+02$	0		$6.07E-03$	$2.83E-03$	0	
PSO-cf-local	$5.26E+01$	$1.43E+01$	0		$5.96E+02$	$3.86E+02$	0		$6.84E-03$	$2.05E-03$	0	
FIPS-PSO	$1.75E+02$	$8.79E+00$	0		$1.47E+03$	$5.40E+02$	0		$8.61E-03$	$2.12E-03$	0	
CPSO-H	$3.77E+02$	$1.10E+02$	0		$2.54E+04$	$1.15E+04$	0		$5.30E-02$	$2.17E-02$	0	
CLPSO	$1.03E+02$	$1.26E+01$	0		$6.96E+03$	$1.49E+03$	0		$1.86E-02$	$7.92E-03$	0	
SLPSO	$3.15E+01$	$6.41E+00$	0		$7.79E-04$	$2.24E-03$	7		$2.43E-03$	$7.71E-04$	0	
PSO-IVL	$\mathbf{0.00E+00}$	$\mathbf{0.00E+00}$	**30**	**116,660**	$\mathbf{0.00E+00}$	$\mathbf{0.00E+00}$	**30**	**102,080**	$\mathbf{7.55E-05}$	$\mathbf{7.51E-08}$	**0**	

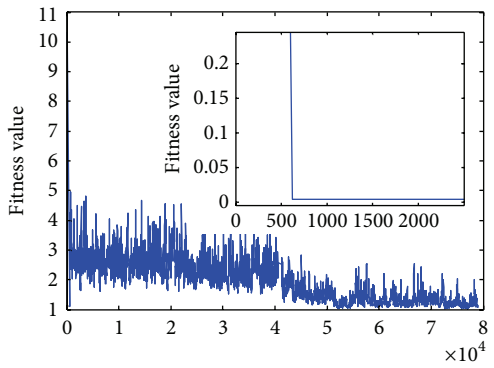


(a) Sphere function

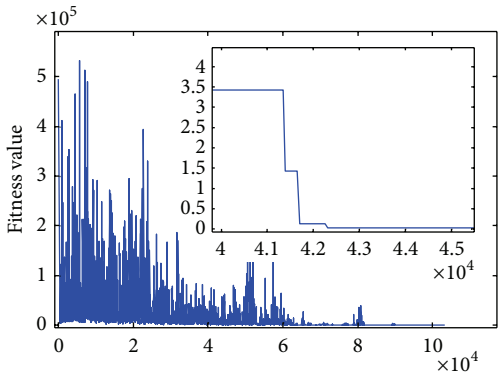
(b) Rastrigin function



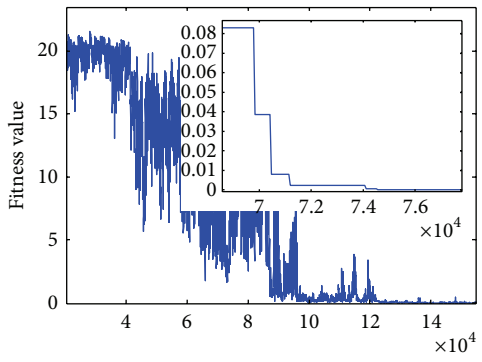
(c) Rosenbrock function



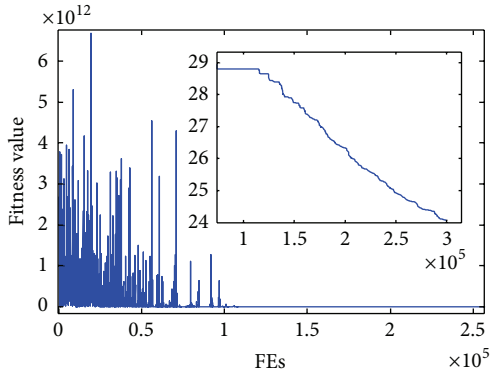
(d) Griewank function



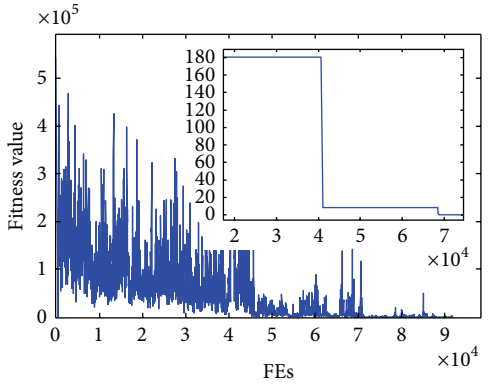
(e) Schwefel 1.2 function



(f) Ackley function

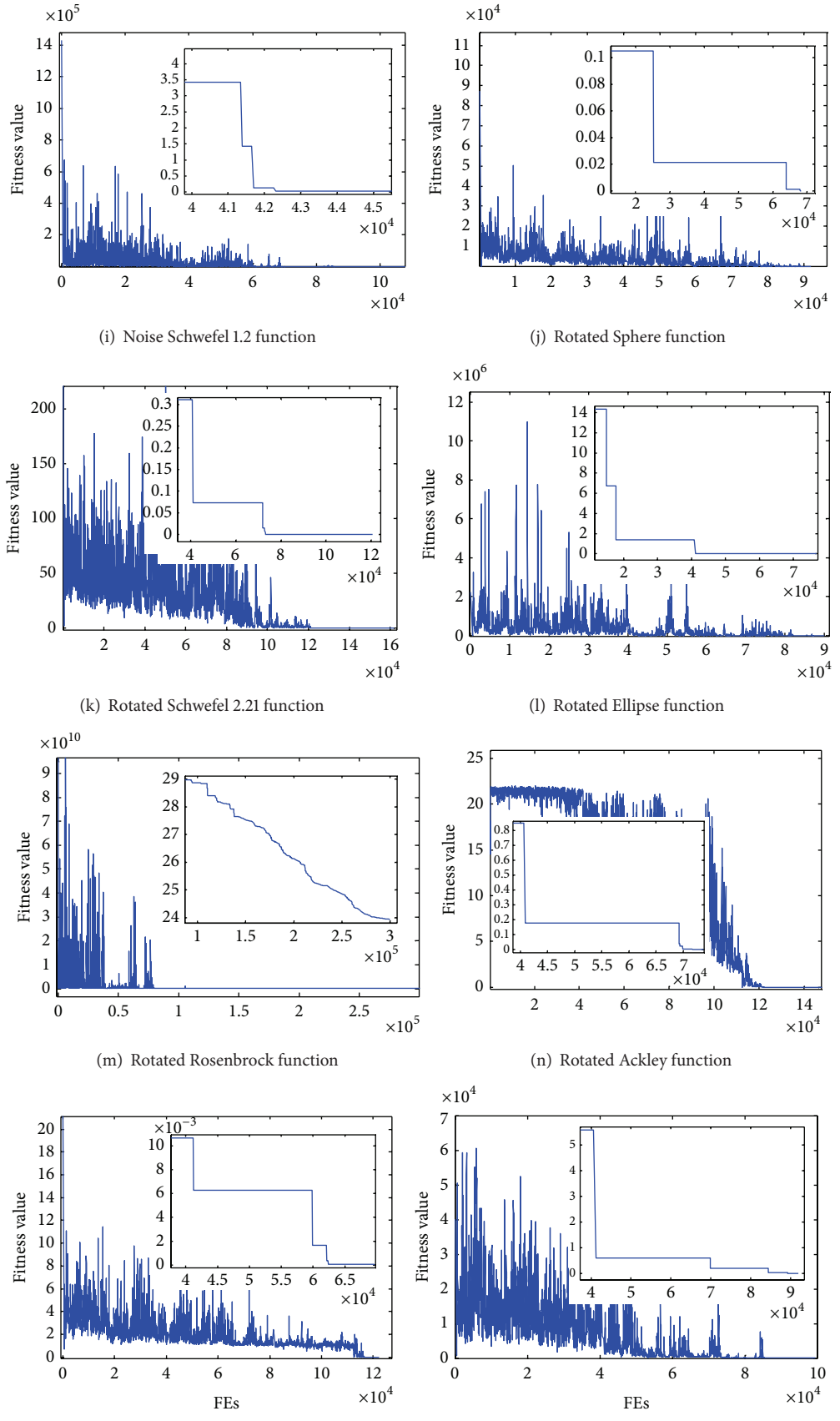


(g) Scaled Rosenbrock 100 function



(h) Scaled Rastrigin 10 function

Figure 1: Continued.



(i) Noise Schwefel 1.2 function

(j) Rotated Sphere function

(k) Rotated Schwefel 2.21 function

(l) Rotated Ellipse function

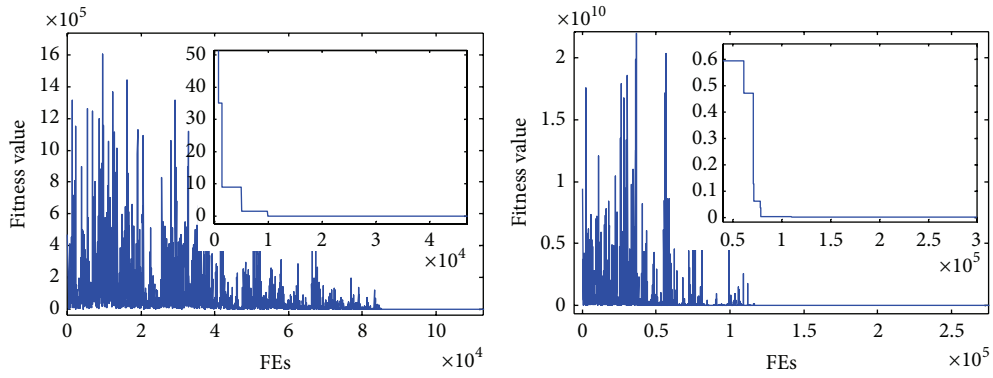
(m) Rotated Rosenbrock function

(n) Rotated Ackley function

(o) Rotated Griewank function

(p) Rotated Rastrigin function

Figure 1: Continued.



(q) Noise Rotated Schwefel 1.2 function (r) Noise Rotated Quadric function

Figure 1: Evolutionary process of a particle and the global best solution with regard to each test function.

its overall evolution. We enlarge and display the evolutionary process of global-best-so-far solution at some stages.

Two observations are worth making here. First, the fitness of the particle fluctuates quite substantially at the early stage of PSO-IVL but gradually diminishes and finally converges to the optimal solution. This is because of the trap detection and jumping out strategy adopted in PSO-IVL. During the evolutionary process, if the particle is detected to be trapped in a local optimum, the particle performs Gaussian mutation, which adaptively enables the particle to randomly move to a new position. The adaptive Gaussian mutation operator makes the particle fluctuate to a large extent in the early stage of optimization. In addition, the learning and interaction strategies realized within the swarm enable the particle to always converge to a good solution.

Second, the particle exhibits a certain probability to determine high-quality solutions at the early stage. The reason is that PSO-IVL employs the IVL strategy, which enables the particle to learn among different variables. As a result, just a good value of a variable can quickly lead the particle to reach a position with good fitness.

The smaller figures indicate that PSO-IVL can converge to a high-quality solution fast on each test function and finally find the optimal solutions for most functions. This can be explained by the fact that the IVL strategy indeed enables the particle to find high-quality position at high speed and high probability; meanwhile, the trap detection and jumping out strategy can help particles escape from local optima. As a conclusion, the combination of the IVL strategy, the trap detection and jumping out strategy, and the basic PSO forms an efficient optimization environment.

5.4. Analysis of the Impact of the Inner Variable Learning Strategy. As we know, some optimization functions may be partial symmetric. In this case, when we use PSO-IVL to carry out optimization, only a portion of their variables can be utilized to realize the IVL strategy. Therefore, it is important to investigate the impact of the number of variables being involved in the IVL strategy. For convenient comparison, we selected six complex optimization functions (where it is hard to obtain an optimal or near optimal solution for these functions without using the IVL strategy) and solved them by using PSO-IVL with a different number of variables involved in the IVL strategy. This means that, even though in these functions all variables are symmetric, each time we only set a certain number of variables to run the IVL strategy. The obtained results are listed in Table 3, where n stands for the number of variables executed by the IVL strategy. We set n to 0, 5, 10, 15, 25, and 30, respectively. When n is equal to 0, this means that in fact the IVL strategy is not invoked. When n equals 30, all variables are viewed as symmetric and adopted to execute the IVL strategy.

From the results displayed in Table 3, we can find that, for every selected optimization function, the solution obtained by PSO-IVL is getting better with the increase of the values of n. Therefore, we can come to some conclusions. (1) The effectiveness of the IVL strategy is significant. (2) More variables in an optimization function being utilized the IVL strategy (i.e., more variables are symmetric) will lead to much better solutions. (3) Even if there are only less symmetric variables in an optimization function, the employment of the IVL strategy on these variables has potential to improve the optimization process.

6. Conclusions

In this work, we have introduced a new knowledge-driven PSO variant (PSO-IVL), which integrates the generic PSO, a novel inner variable learning (IVL) strategy, and a novel trap detection and jumping out strategy. The IVL strategy is based on the knowledge that the values of symmetric variables in an optimization function will satisfy certain relations in the optimal solution. The trap detection and jumping out strategy is established at the level of individual particles rather than the swarm level, which improves the flexibility and adaptability of particles and helps particles escape from local optima. Experimental simulations completed for some classical optimization functions demonstrate the competitive performance of PSO-IVL, which is superior to all the selected state-of-the-art peer PSO variants.

Although we choose completely symmetric functions in which all variables are symmetric to test our algorithm's

Table 3: Results of different numbers of variables being executed with the IVL strategy.

Functions	$n = 0$	$n = 5$	$n = 10$	$n = 15$	$n = 20$	$n = 25$	$n = 30$
Scaled Rosenbrock 100: f_7	$8.45E+04$	$2.34E+04$	$5.82E+03$	$3.65E+03$	$1.48E+02$	$7.55E+01$	$2.32E+01$
Noise Schwefel 1.2: f_9	$9.82E+02$	$6.37E+01$	$1.46E+00$	$8.17E-04$	$3.25E-08$	$9.72E-13$	$0.00E+00$
Rotated Schwefel 2.21: f_{11}	$6.63E-01$	$1.16E-01$	$3.61E-02$	$8.28E-04$	$2.52E-08$	$1.07E-14$	$0.00E+00$
Rotated Ellipse: f_{12}	$2.91E+02$	$3.97E+00$	$1.76E-03$	$5.11E-06$	$5.34E-11$	$6.28E-19$	$0.00E+00$
Rotated Rosenbrock: f_{13}	$1.82E+04$	$8.82E+03$	$2.57E+03$	$9.07E+02$	$2.93E+02$	$5.64E+01$	$2.87E+01$
Noise Rotated Schwefel1.2: f_{17}	$2.77E+03$	$5.75E+02$	$6.03E+00$	$4.23E-04$	$2.45E-09$	$1.87E-15$	$0.00E+00$

performance, the proposed algorithm can also be applied to partial symmetric functions (in which only some variables are symmetric). In this case, we just need to let the IVL strategy be performed on the symmetric variables. Moreover, the IVL strategy can be integrated into existing PSO alternatives.

PSO-IVL will be effective in optimization functions possessing symmetric variables. However, it is meaningful for three reasons. Firstly, it can obtain good solutions (usually the optimal solutions) for many benchmark functions. Secondly and more importantly, the efficiency of PSO-IVL indicates that the combination of the problem-oriented knowledge and PSO would be a promising direction for applying PSO to optimization problems. Thirdly, since symmetry is a general phenomenon existing in nature and engineering, it could be beneficial to check the variable symmetry when we try to use PSO or other evolutionary algorithms to solve a new complex optimization problem.

The future research can be carried out in three directions. One can look at discovering and formalizing domain knowledge (e.g., generic quantitative relations among different variables) existing in optimization problems and integrate it into the design of more advanced PSO schemes. The second one is to attempt to apply PSO-IVL to some real-life optimization problems. The third direction could be to formulate a general framework for guiding knowledge discovery in optimization problems and its integration into evolutionary algorithms.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work was supported by the National Nature Science Foundation of China (NSFC, 71271213, 51178193, and 41001220). The author Guohua Wu is supported by the China Scholarship Council under Grant no. 201206110082.

References

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

[2] 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, October 1995.

[3] R. Eberhart, Y. Shi, and J. Kennedy, *Swarm Intelligence*, Morgan Kaufmann, 2001.

[4] 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.

[5] R. Eberhart and Y. Shi, "Comparison between genetic algorithms and particle swarm optimization," in *Evolutionary Programming VII*, pp. 611–616, 1998.

[6] Y. Shi and R. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII*, pp. 591–600, 1998.

[7] A. Chatterjee and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," *Computers & Operations Research*, vol. 33, no. 3, pp. 859–871, 2006.

[8] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.

[9] Y. Shi and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 101–106, May 2001.

[10] A. Ismail and A. Engelbrecht, "The self-adaptive comprehensive learning particle swarm optimizer," *Swarm Intelligence*, pp. 156–167, 2012.

[11] K. E. Parsopoulos and M. N. Vrahatis, "Parameter selection and adaptation in unified particle swarm optimization," *Mathematical and Computer Modelling*, vol. 46, no. 1-2, pp. 198–213, 2007.

[12] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation*, 1999.

[13] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," Proceedings of the Congress on Evolutionary Computation, 1999.

[14] X. Hu and R. Eberhart, "Multiobjective optimization using dynamic neighborhood particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1677–1681, 2002.

[15] J. J. Liang and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 124–129, June 2005.

[16] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1671–1676, 2002.

[17] 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.

[18] M. Hu, T. Wu, and J. D. Weir, "An intelligent augmentation of particle swarm optimization with multiple adaptive methods," *Information Sciences*, vol. 213, pp. 68–83, 2012.

[19] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 42, pp. 627–646, 2012.

[20] P. Angeline, "Evolutionary optimization versus particle swarm optimization: philosophy and performance differences," in *Evolutionary Programming VII*, pp. 601–610, 1998.

[21] C. Wei, Z. He, Y. Zhang, and W. Pei, "Swarm directions embedded in fast evolutionary programming," in *Proceedings of the Congress on Evolutionary Computation*, pp. 1278–1283, 2002.

[22] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[23] R. Poli, C. Di Chio, and W. B. Langdon, "Exploring extended particle swarms: a genetic programming approach," in *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 169–176, June 2005.

[24] Y. Wang, B. Li, T. Weise, J. Wang, B. Yuan, and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4515–4538, 2011.

[25] L.-N. Xing, Y.-W. Chen, P. Wang, Q.-S. Zhao, and J. Xiong, "A knowledge-based ant colony optimization for flexible job shop scheduling problems," *Applied Soft Computing Journal*, vol. 10, no. 3, pp. 888–896, 2010.

[26] G. Wu, J. Liu, M. Ma, and D. Qiu, "A two-phase scheduling method with the consideration of task clustering for earth observing satellites," *Computers & Operations Research*, vol. 40, pp. 1884–1894, 2013.

[27] G. Wu, M. Ma, J. Zhu, and D. Qiu, "Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks," *Journal of Systems Engineering and Electronics*, vol. 23, pp. 723–733, 2012.

[28] H. Li and B. Wu, "Adaptive geo-information processing service evolution: reuse and local modification method," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 83, pp. 165–183, 2013.

[29] H. Li, Q. Zhu, X. Yang, and L. Xu, "Geo-information processing service composition for concurrent tasks: a QoS-aware game theory approach," *Computers & Geosciences*, vol. 42, pp. 46–59, 2012.

[30] G. Wu, W. Pedrycz, H. Li, D. Qiu, M. Ma, and J. Liu, "Complexity reduction in the use of evolutionary algorithms to function optimization: a variable reduction strategy," *The Scientific World Journal*, vol. 2013, Article ID 172193, 8 pages, 2013.

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

[32] 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.

[33] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[34] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[35] 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.

[36] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization—part I: background and development," *Natural Computing*, vol. 6, no. 4, pp. 467–484, 2007.

[37] A. Banks, J. Vincent, and C. Anyakoha, "A review of particle swarm optimization—part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications," *Natural Computing*, vol. 7, no. 1, pp. 109–124, 2008.

[38] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence*, vol. 1, pp. 33–57, 2007.

[39] Y. del Valle, G. K. Venayagamoorthy, S. Mohagheghi, J.-C. Hernandez, and R. G. Harley, "Particle swarm optimization: basic concepts, variants and applications in power systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171–195, 2008.

[40] S. Rana, S. Jasola, and R. Kumar, "A review on particle swarm optimization algorithms and their applications to data clustering," *Artificial Intelligence Review*, vol. 35, no. 3, pp. 211–222, 2011.

[41] D. W. Boeringer and D. H. Werner, "Particle swarm optimization versus genetic algorithms for phased array synthesis," *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 3, pp. 771–779, 2004.

[42] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, 1999.

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

[44] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 34, no. 2, pp. 997–1006, 2004.

[45] Y.-P. Chen, W.-C. Peng, and M.-C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 37, no. 6, pp. 1460–1470, 2007.

[46] P. S. Andrews, "An investigation into mutation operators for particle swarm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '06)*, pp. 1044–1051, July 2006.

[47] W.-J. Zhang and X.-F. Xie, "DEPSO: hybrid particle swarm with differential evolution operator," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 3816–3821, October 2003.

[48] M. G. H. Omran, A. P. Engelbrecht, and A. Salman, "Differential evolution based particle swarm optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 112–119, April 2007.

[49] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[50] P. N. Suganthan, N. Hansen, J. J. Liang et al., *Problem Definitions and Evaluation Criteria for the CEC, 2005 Special Session on Real-Parameter Optimization*, Nanyang Technological University, Singapore, 2005.

[51] C.-Y. Chen, K.-C. Chang, and S.-H. Ho, "Improved framework for particle swarm optimization: swarm intelligence with diversity-guided random walking," *Expert Systems with Applications*, vol. 38, no. 10, pp. 12214–12220, 2011.