Expert Systems with Applications 41 (2014) 7536–7548

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Superior solution guided particle swarm optimization combined with local search techniques

Guohua Wu [a], Dishan Qiu [a], Ying Yu [b], Witold Pedrycz [c,d], Manhao Ma [a], Haifeng Li [e,*]

[a] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, Hunan, PR China
[b] School of Software, Jiangxi Agricultural University, Nanchang 330045, Jiangxi, PR China
[c] Department of Electrical & Computer Engineering, University of Alberta, Edmonton AB T6R 2V4, Canada
[d] Warsaw School of Information Technology, Newelska 6, Warsaw, Poland
[e] School of Civil Engineering and Architecture, Central South University, Changsha 410004, Hunan, PR China

ARTICLE INFO

Article history:
Available online 11 June 2014

Keywords:
Particle swarm optimization
Evolutionary algorithm
Individual level based mutation
Local search
Gradient-based local search
Derivative-free local search

ABSTRACT

Particle swarm optimization (PSO) is an evolutionary algorithm known for its simplicity and effectiveness in solving various optimization problems. PSO should have strong yet balanced exploration and exploitation capabilities to enhance its performance. A superior solution guided PSO (SSG-PSO) framework integrated with an individual level based mutation operator and different local search techniques is proposed in this study. In SSG-PSO, a collection of superior solutions is maintained and updated with the evolutionary process, such that each particle can comprehensively learn from the recorded superior solutions. In addition, to maintain the diversity of the particle swarm, SSG-PSO is combined with an individual level based mutation operator, which will be invoked when a particle is trapped in a local optimum (determined by the fitness and position states of the particle), thereby improving the adaptation and flexibility of each individual particle. Moreover, two gradient-based local search techniques, namely, the Broyden–Fletcher–Goldfarb–Shanno (BFGS) and Davidon–Fletcher–Powell (DFP) Quasi–Newton methods, and two derivative-free local search techniques, namely, pattern search and Nelder–Mead simplex search, are incorporated into SSG-PSO. The performances of SSG-PSO and that of its local search enhanced variants are extensively and comparatively studied on a suit of benchmark optimization functions.

1. Introduction

Optimization is important in scientific research, management, and industry because numerous real-world problems can be essentially modelled as optimization tasks. "Traditional" mathematical programming methods (e.g., gradient-based methods) are no longer completely effective in solving complex optimization problems characterised by multi-modality, discontinuity, and high dimensionality. Thus, many evolutionary algorithms (EAs), such as genetic algorithm (GA) and ant colony optimization (ACO), have emerged. Particle swarm optimization (PSO), developed by Kennedy and Eberhart (Eberhart & Kennedy, 1995; Kennedy & Eberhart, 1995), is a competitive population-based algorithm, which is particularly efficient in dealing with numerical optimization problems. PSO is a bio-inspired (Eberhart, Shi, & Kennedy, 2001) algorithm which mimics swarm behaviours, such as bird flocking and fish schooling (Zhan, Zhang, Li, & Shi, 2011). Particles in PSO adjust their movements by learning from their own past experience and that of their neighbours in an attempt to identify better positions in a cooperative manner.

PSO has attracted significant attention since its inception because of its simplicity and effectiveness in solving various optimization problems. Various strategies, including parameter tuning (Chatterjee & Siarry, 2006; Ismail & Engelbrecht, 2012; Parsopoulos & Vrahatis, 2007; Ratnaweera, Halgamuge, & Watson, 2004; Shi & Eberhart, 1998b; Shi & Eberhart, 2001), topology structure adjustment (Hu & Eberhart, 2002; Kennedy, 1999; Kennedy & Mendes, 2002; Liang & Suganthan, 2005; Suganthan, 1999), intelligent combination of various search strategies (Li, Yang, & Nguyen, 2012; Hu, Wu, & Weir, 2012; Zhan, Zhang, Li, & Chung, 2009), and hybridisation with other optimization techniques (Angeline, 1998; Poli, Di Chio, & Langdon, 2005; Wei, He, Zhang, & Pei, 2002; Yao, Liu, & Lin, 1999), have recently been developed to strengthen the performance of PSO (Wu et al., 2014). Although noticeable progress and fruitful achievements have been attained, successfully balancing the exploration and exploitation capabilities of PSO to determine high-quality solutions for complex optimization problems, especially those with multimodal landscapes or high-relevance variables, remains a fundamental challenge.

* Corresponding author. Tel.: +86 15608418565.
E-mail address: lihaifeng@csu.edu.cn (H. Li).

http://dx.doi.org/10.1016/j.eswa.2014.06.005
0957-4174/

Exploration and exploitation capabilities are two key aspects in the design of efficient EAs (Das & Suganthan, 2011). However, balancing the exploration and exploitation capabilities of an EA is challenging because these capabilities generally contradict each other. Three methods are used to obtain an exploration- and exploitation-balanced EA. The first is integrating diversification strategies (e.g., utilisation of mutation strategies) into an EA which suffers from premature convergence (Andrews, 2006; Gao & Xu, 2011; Pehlivanoglu, 2013). The second is dynamically adjusting the parameters of EA in accordance with evolutionary states (Ismail & Engelbrecht, 2012; Zhan et al., 2009). The third is designing effective exploration and exploitation strategies and then reasonably organising them into an EA.

Our work in this study conforms to the third method. A superior solution guided particle swarm optimization with local search techniques is proposed in this study. The newly presented algorithm consists of three parts, namely, a superior solution guided PSO as framework, an individual level based mutation strategy, and different local search techniques.

Two strategies are adopted in the superior solution guided PSO framework. First, a collection of superior solutions was maintained as learning sources for each particle. Similar techniques, such as niching (Qu, Liang, & Suganthan, 2012; Li, 2010) and example particles (Huang, Qin, Hao, & Lim, 2012), have demonstrated effectiveness in enabling particles to have high exploration capability. The superior solution collection in this study is composed of two solutions; the local best particle solutions and the solutions with high-quality objective function values. Second, each particle comprehensively learns from all recorded superior solutions. Comprehensive learning is particularly effective in dealing with multimodal optimization problems (Liang, Qin, Suganthan, & Baskar, 2006).

An individual level based mutation operator is incorporated into the superior solution guided PSO framework. Mutation operators, such as Gaussian mutation, Cauchy Mutation, Lévy Mutation and Michalewicz's non-uniform mutation, are useful in maintaining the diversity of the swarm, thereby preventing PSO from premature convergence (Gao & Xu, 2011; Wang, Wang, & Wu, 2013; Andrews, 2006; Pehlivanoglu, 2013). Previously, mutation operators have been conducted based on population level, i.e., the whole population is mutated with a certain probability (Andrews, 2006), or the mutation operators are triggered according to the fitness value changes of the local best or global best solutions (Wang et al., 2008; Krohling, 2005; Wang et al., 2013). However, shortcomings are still encountered. First, probability based mutation operations without considering the evolutionary states of particles may destroy the consistence of the search behaviours of particles because a particle may be forced to be mutated although it is not trapped or moving to the better solution area. Second, mutation operations in terms of fitness changes of local best or global best solutions also have a problem in which fitness stagnation does not necessarily mean that particles get trapped because particles may still be exploring the solution space.

In our study, mutation is performed at the individual level and trigged by the satisfaction of two conditions. One condition is that a particle does not improve its fitness in a certain number of generations. The second condition is that the movements of the particle are restricted in a small area. That is, the mean distance between the current position of the particle and its previous certain number of positions is smaller than a threshold value. When both conditions are satisfied, the particle is trapped and should execute the mutation operator. The mutation strategy presented in this study can provide more accurate guidance for the diversification of particles while improving the adaptation and flexibility of each single particle.

Different local search techniques are combined into the superior solution guided PSO framework and comparatively studied to enhance the performance of the algorithm. However, compared with conventional mathematical programming (MP) methods, PSO has global search capability. Therefore, it is a natural idea to reasonably combine PSO with traditional MP methods to solve optimization problems to benefit from both the exploration capability of PSO and exploitation capability of MP. The combination of local search techniques and evolutionary algorithms has been investigated in some papers. For example, the gradient-based local search method (Hu et al., 2012; Noel, 2012; Plevris & Papadrakakis, 2011; Xie, Yu, & Zou, 2012) and derivative-free local search method (Fan & Zahara, 2007; Qu et al., 2012) have been employed in EAs to provide stronger exploitation capability. However, these methods have not been comprehensively and comparatively studied yet. In the present study, four gradient-based and derivative-free local search techniques, including the Broyden–Fletcher–Goldfarb–Shanno (BFGS) and Davidon–Fletcher–Powell (DFP) Quasi-Newton methods (gradient-based), Pattern Search (derivative-free), and Nelder–Mead simplex search (derivative-free) method are integrated into the algorithm, respectively.

The paper is structured as follows: Section 2 briefly introduces the basic PSO and reviews the related work. Section 3 describes the superior solutions guided PSO framework. Section 4 briefly introduces the four adopted local search techniques. Section 5 provides details of the proposed individual level based mutation operator. Section 6 provides the algorithm framework of SSG-PSO with different local search techniques. Section 7 conducts experimental and comparative studies on a suit of 21 benchmark optimization problems. Section 8 provides the conclusion of this paper.

2. Related studies

PSO has undergone a significant progress since its introduction in 1995. A large number of PSO variants have been proposed to improve the performance of traditional PSO (Wu et al., 2014). Comprehensive reviews of PSO can be found in (Banks, Vincent, & Anyakoha, 2007; Banks, Vincent, & Anyakoha, 2008; Poli, Kennedy, & Blackwell, 2007). In addition, del Valle, Venayagamoorthy, Mohagheghi, Hernandez, and Harley (2008) surveyed PSO along with its basic concepts, variants, and applications in power systems. Rana, Jasola, and Kumar (2011) reviewed PSO and its application to data clustering. In this section, this study briefly introduces the basic PSO, and then surveys the recent major PSO variants.

2.1. Basic PSO

Analogous to other evolutionary algorithms, such as GA and ACO, PSO is a population-based stochastic optimization algorithm. A swarm of particles attempt to search for superior solutions through learning, communication and interaction. The position of each particle refers to a solution. The position moving process of a particle in the solution space then 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 is the number of variables encountered in the optimization problem. In the generic PSO with inertia weight (Shi & Eberhart, 1998a), the position and velocity of particle i are updated during the evolutionary process:

$$v_i^d = w \times v_i^d + c_1 \times r_1^d \times (pBest_i^d - x_i^d) + c_2 \times r_2^d \times (gBest^d - x_i^d), \quad (1)$$

$$x_i^d = x_i^d + v_i^d, \quad (2)$$

where x_i^d is the dth variable (or dimension) of the position of particle i; v_i^d is the dth variable of the velocity of particle i; $pBest_i^d$ is the

dth variable of the personal historical best position found by particle i; $gBest_d$ is the dth variable of the global best position found by the whole swarm; c_1 and c_2 are the acceleration parameters commonly set to 2.0; r_1^d and r_2^d are two random numbers drawn from a uniform distribution over [0, 1] and w is the inertia weight 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 (Shi & Eberhart, 1998a).

The behaviour of the particle is specified by its velocity and position update in (1) and (2) (Boeringer & Werner, 2004; del Valle et al., 2008). The first inertia weight component of (1) models the tendency of the particle to continue in the same direction. The second component of (1) is the particle "memory", "self-knowledge", "nostalgia" or "remembrance" (Boeringer & Werner, 2004; del Valle et al., 2008), which reflects the self-learning behaviour of the particle. The third component in (1) is the "cooperation", "social knowledge", "group knowledge" or "shared information" (Boeringer & Werner, 2004; del Valle et al., 2008), which reflects the social learning behaviour of the particle. Eq. (2) indicates that the position of the particle changes in terms of its current position and velocity.

2.2. Important PSO variants

"Standard" PSO exhibits some deficiencies, including premature convergence and inefficiency in solving complex multi-modal optimization problems (Wu et al., 2013). One way to strengthen the capability of PSO is to dynamically adapt its parameters when running the particle evolutionary process. For instance, the inertia weight parameter w is set to linearly decrease over iterations (Shi & Eberhart, 1998a; Shi & Eberhart, 1999). In addition, a fuzzy adaptive mechanism is used to tune the value of w (Shi & Eberhart, 2001). Kennedy and Eberhart recommended that the proper value for the acceleration parameters c_1 and c_2 can be fixed and set to 2.0 (Eberhart & Kennedy, 1995). These value settings were accepted in many works. In comparison, Suganthan suggested that the use of ad hoc selected values for c_1 and c_2 rather than the fixed values for different problems can result in better performance (Suganthan, 1999). Ratnaweera et al. (2004) presented a PSO variant with linearly time-varying acceleration coefficients (HPSO-TVAC). Zhan et al. (2009) 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, i.e., exploration, exploitation, convergence and jumping out state. Ismail and Engelbrecht (2012) controlled the parameters of PSO by embedding them into the position vectors of particles, which enhanced the performance of the comprehensive learning PSO variant (CLPSO) (Liang et al., 2006).

In addition to parameter adaptation, topological structures of the particle swarm are extensively studied. For example, Kennedy (1999), Kennedy and Mendes (2002) suggested that a small neighbourhood may be more suitable for complicated multimodal problems, whereas a larger neighbourhood may be more effective for unimodal problems. Kennedy and Mendes (Kennedy & Mendes, 2002) evaluated 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 with the others. However, the selection of an appropriate neighbourhood structure is problem-dependent. Being aware of the noticeable effect of neighbourhood structures, the neighbourhood structure dynamic adaptation mechanisms have been investigated by researchers (Kennedy, 1997; Suganthan, 1999). Mendes, Kennedy, and Neves (2004) presented a fully informed particle swarm (FIPS) in which each individual learns the experience of all its neighbours rather than just the best one and itself. In Maruta, Kim, Song, and Sugie (2013), a cyclic neighbourhood is employed in PSO. Qu, Suganthan, and Das (2013) proposed a distance-based locally informed particle swarm (LIPS) optimiser, which eliminates the need to specify any niching parameter and enhances the fine search ability of PSO.

Another natural evolution of the PSO can be achieved by incorporating operators or techniques effectively used in other evolutionary algorithms (del Valle et al., 2008). Angeline (Angeline, 1998) developed a hybrid PSO by introducing the selection operator from GA. A hybrid PSO based on genetic programming (GP) was presented by Poli et al. (2005). In Juang (2004), Juang integrated GA with PSO for designing artificial neural network. Gholizadeh (2013) combined cellular automata with PSO to deal with layout optimization of truss structures. Other operations and techniques, such as crossover (Chen, Peng, & Jian, 2007), mutation (Andrews, 2006), variable neighbourhood search (Costa, Goldbarg, & Goldbarg, 2012), and differential evolution (Omran, Engelbrecht, & Salman, 2007; Zhang & Xie, 2003) have been combined into PSO as well.

An intelligent ensemble of different learning strategies in the swarm evolutionary process is a promising direction for designing efficient PSO variants. A collection of learning strategies with different capabilities, such as exploitation, exploration or jumping out from local optima, is usually prepared, and then enables each particle to automatically choose learning strategies through a sophisticated adaptation mechanism to determine its next move. Many state-of-the-art PSO variants have been developed following this design principle. Liang et al. (2006) proposed a comprehensive learning PSO (CLPSO), which uses a novel learning strategy whereby all other particle historical best information is used to update a particle velocity. In Zhan et al. (2011) proposed an orthogonal learning (OL) strategy for PSO to discover more useful information which lies in two particle experiences through orthogonal experimental design. Experimental results demonstrated that OLPSO significantly improves the performance of PSO, offering faster convergence, higher solution quality and better robustness. Hu et al. (2012) proposed a PSO variant with intelligent combination of a non-uniform mutation-based method and an adaptive sub-gradient method. A Cauchy mutation operator was further utilised to prevent premature convergence. Li et al. (2012) presented a self-learning PSO, in which each particle has four strategies to cope with different situations confronted in the search process. In Chen et al. (2013), an aging leader and challenger mechanism has been proposed to adaptively establish a suitable leader to lead the swarm. Gholizadeh and Moghadas (2014), Gholizadeh, Torkzadeh, and Jabarzadeh (2013) improved the performance of traditional PSO by introducing the quantum behaviour-based learning strategy.

The combination of traditional mathematical programming (MP) techniques and EAs has recently started to attract the attention of researchers. MP techniques are more powerful in local search. That is, they can converge to a local optimum more efficiently. Compared with MP, EAs have better exploration capability. To enhance the exploitation capability of evolutionary algorithms, different MP techniques are employed as local search strategies of EAs (Noel, 2012; Plevris & Papadrakakis, 2011; Qu et al., 2012). These local search strategies are expected to support EAs to more efficiently find a local optimum, or a global optimum when EAs have already located the global optimal basin. Both gradient-based local search (Noel, 2012; Plevris & Papadrakakis, 2011; Xie et al., 2012) and derivative-free local search techniques (Fan & Zahara, 2007; Qu et al., 2012) have been adopted to date. In Fan and Zahara (2007), Nelder–Mead simplex search method has been combined with PSO and showed its effectiveness in improving the efficiency of PSO. The gradient information can be viewed as

a problem domain knowledge which has attracted increasing attention when using EAs to deal with concrete optimization problems. For discrete optimisation, using ACO as an example, the domain knowledge was used for solution representation and neighbourhood search (Wu, Liu, Ma, & Qiu, 2013; Wu, Ma, Zhu, & Qiu, 2012). With regard to continuous optimisation, knowledge-based variable symmetry and variable reduction strategies have been investigated (Wu, Pedrycz, et al., 2013; Wu et al., 2014).

Notably, in addition to the combination of PSO with MP local search techniques, researchers also have investigated the integration of PSO with other meta-heuristics which function as local search methods, such as Ant Colony Optimization (ACO), Harmony Search (HS) and firefly Algorithm (FA) (Kaveh & Talatahari, 2009a; Kaveh & Talatahari, 2009b). Related experimental results showed that hybridisation of PSO and such meta-heuristics can also noticeably enhance the performance of traditional PSO.

3. Superior solutions guided particle swarm optimisation

In the original PSO, given that each particle learns from its local best and the global best solutions simultaneously, particles may get trapped in a local optimum soon when the current global best solution is a local optimum and not updated in a certain number of generations. In a continuous solution space, high quality solutions share similarities with the theoretical optimal solution. Therefore, two mechanisms can be useful to improve the performance of PSO. One is to maintain a certain number of diversified yet high-quality solutions. The second is to enable particles to comprehensively learn from high-quality solutions (superior solutions). The comprehensive learning from multiple local best solutions has gained success in solving multi-modal optimization problems (Liang et al., 2006).

The superior solution set is composed of two solutions, i.e., local best solutions and other high-quality solutions. Let S denote the superior solution set; S_1 denotes the set of all local best solutions; S_2 includes other solutions with satisfactory objective function values. On the one hand, each particle at each generation will update the local best solution when a better solution has been found by the particle and the former local best solution is discarded accordingly. However, the former local best solution to be discarded may still have high quality and is not located in the same basin as the newly found local best solution. On the other hand, each particle at each generation will discard its current position when it is not better than its local best solution. Similarly, the current position can also have relatively high quality and contain promising information related to the optimal solution. The additional collection S_2 is used to record these potential good solutions, thereby enriching the pool of superior solutions.

The local best solution $pBest_i$ of particle i at each generation will be modified or not. Therefore, either the current local best solution $pBest_i$ or current position pos_i will be traditionally discarded. This work evaluated whether $pBest_i$ or pos_i is qualified to replace the worst solution in S_2 in the case where either $pBest_i$ or pos_i has a better objective function value.

Each particle learns from its local best solution and other superior solutions in S, $S = S_1 \cup S_2$. For total D dimensions, m dimensions were randomly selected for each particle i to learn from other superior solutions in S and other $D - m$ dimensions learn from its local best solution. Let M denote the collection of m randomly selected dimensions. For each dimension d in M, the roulette wheel selection procedure is used to choose a superior solution $pR_{f(d)}$ different from $pBest_i$ in S. The procedure for selecting $pR_{f(d)}$ is that, for the dimension d in M, two solutions (different from $pBest_i$) were randomly selected from S. The one with better objective function value is then assigned to $pR_{f(d)}$.

The velocity update of particle i will be updated as follows:

$$v_i^d \leftarrow \begin{cases} w \times v_i^d + c \times r_1^d \times (pBest_i^d - x_i^d) & \text{if } d \notin M \\ w \times v_i^d + c \times r_1^d \times (pR_{f(d)}^d - x_i^d) & pR_{f(d)} \in S,\ pR_{f(d)} \neq pBest_i \text{ otherwise} \end{cases} \tag{3}$$

After updating the velocity, the position of particle i is modified according to (2).

4. Local search methods

4.1. BFGS and DFP algorithm

Conventional Newton's methods use the gradient and the Hessian matrix of second derivatives of the optimization function to be minimised (Sun & Yuan, 2006). In addition, in Newton's methods, the Hessian matrix is required to be calculated and inverted, which is usually quite costly in computing. By contrast, in Quasi-Newton methods, the conversion of Hessian is updated by analysing successive gradient vectors.

By measuring the changes in gradients, Quasi-Newton methods construct a model of the objective function which is good enough to produce super-linear convergence. The improvement over steepest descent is dramatic, especially on difficult problems. Moreover, given that the second derivatives are not required, Quasi-Newton methods are sometimes more efficient than the Newton's method (Nocedal & Wright, 2000). Various Quasi-Newton methods have emerged; among which the well-known ones include SR1 formula (symmetric rank one), Broyden class, Broyden–Fletcher–Goldfarb–Shanno (BFGS) method and Davidon–Fletcher–Powell (DFP) algorithm. In this study, the performance of BFGS and DFP was investigated when they are combined into PSO.

The framework of BFGS and DFP can be described in Algorithm 1 (Nocedal & Wright, 2000).

Algorithm 1: BFGS and DFP algorithm framework

Given starting point x_0, convergence tolerance ε, initial inverse Hessian approximation H_0;
$k = 0$;
// ∇f_k *is the gradient of the objective function*
While $||\nabla f_k|| > \varepsilon$
  Compute search direction $p_k = -H_k \nabla f_k$;
  Set $x_{k+1} = x_k + \alpha_k p_k$, where α_k is the step and computed from line search method;
  Calculate $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$;
  Compute H_{k+1} by H_k, s_k and y_k according to (4) or (5);
  $k = k + 1$;
End while

The difference between BFGS and DFP is the way for updating the inverse Hessian approximation at each iterate. In BFGS, H_{k+1} is calculated as follows:

$$H_{k+1} = (I - p_k s_k y_k^T) H_k (I - p_k y_k s_k^T) + p_k s_k s_k^T. \tag{4}$$

While in DFP, H_{k+1} is computed as follows:

$$H_{k+1} = H_k - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k} + \frac{s_k s_k^T}{y_k^T s_k}. \tag{5}$$

Given that some functions can be too complex to easily obtain an analytical gradient or not differential everywhere, in real application the numerical gradient is usually utilised instead (Bräuninger, 1981).

4.2. Pattern search and Nelder–Mead simplex search

In the real-world, problems in which derivatives are not available may be encountered. In this case, derivative-free optimization

algorithms can be promising alternative methods. Nelder–Mead simplex search method and Pattern Search are two popular derivative-free optimization algorithms and are employed in this study. Nelder–Mead simplex search and its integration with PSO have been introduced in Fan and Zahara (2007). A brief description of Pattern Search method is given in this study.

Notations are given to define pattern-search methods. For the current iterate, x_k is a solution, D_k denotes the set of possible search directions and γ_k is the parameter of search step. The frame consists of points $x_k + \gamma_k p_k$, for all $p_k \in D_k$. When one of the points in the frame generates significantly better objective function value of f, the step is accepted and may also increase γ_k to expand the frame for the next generation. When none of the points in the frame has significantly better function value than f_k, reduce γ_k, set $x_{k+1} = x_k$, and then repeated. In either case, the direction set D_k may be changed prior to the next iteration and subjected to certain restrictions. The framework of Pattern Search is given in Algorithm 2 (Nocedal & Wright, 2000).

Algorithm 2: Pattern Search framework

Given convergence tolerance γ_{toc}, contraction parameter $\theta_{\max}$;
Sufficient decrease function ρ:$[0, \infty) \to R$ with $\rho(t)$ an increasing function of t and $\rho(t)/t \to 0$, as $t \downarrow 0$;
Choose initial point x_0, initial step length $\gamma_0 > \gamma_{toc}$, initial direction set D_0;
For $k = 1 \to MaxIter$
  If $\gamma_k < \gamma_{toc}$
    Stop;
  If $f(x_k + \gamma_k p_k) < f(x_k) - \rho(\gamma_k)$ for some $p_k \in D_k$
    Set $x_{k+1} = x_k + \gamma_k p_k$ for some such p_k;
    Set $\gamma_{k+1} = \varphi_k \gamma_k$ for some $\varphi_k \geqslant 1$; // to increase the step length
  Else
    Set $x_{k+1} = x_k$;
    Set $\gamma_{k+1} = \theta_k \gamma_k$, where $0 < \theta_k \leqslant \theta_{\max} < 1$;
  End if
End for

5. Individual level based mutation operator

To improve the performance of PSO, the dynamic tune of particle's behaviours according to run-time evolutionary states has attracted much attention. This tune is usually based on the evaluation of the swarm's distribution. For instance, Zhan et al. (2009) determined four evolutionary states (i.e., exploration, exploitation, convergence and jumping out) according to the statistical position distribution of the swarm. They then conducted the automatic parameter adjustment in terms of these different states. The swarm distribution information was obtained by calculating the mean distance of each particle to all other particles. Chen, Chang, and Ho (2011) also used the distance between particles to evaluate the diversity of PSO in the evolutionary process. They incorporated diversity criterion into the objective function to optimise the optimization problem while guaranteeing the diversity of the overall swarm. In addition, various mutation operators have also been integrated into PSO to maintain the diversity and prevent the premature convergence of PSO (Gao & Xu, 2011; Andrews, 2006; Pehlivanoglu, 2013; Wang et al., 2013).

Current strategies for keeping the diversity of PSO are based on the swarm distribution information, only the fitness change of particles, or probability only without any prior knowledge of particles. In our opinion, mutation strategies based on the swarm distribution may not be good enough for the adaptation and flexibility of a single particle. Although the swarm is not crowded at times according to the empirical diversity evaluation criteria, some of the particles may have actually been trapped in optima. However, mutation strategies based only on fitness change may mislead the particle behaviours. Although a particle does not improve its local best solution within a certain number of generations, it may still be in the exploration state.

In this study, a mutation operator based on the combination of individual position distribution and individual fitness change at recent generations is proposed. Two conditions should be satisfied to trigger the mutation operation of each particle i. The first condition is that the local best solution of particle i has stagnated for certain t number of generations, which can be expressed as follows:

$$sTag_i > t, \tag{6}$$

where $sTag_i$ is the current stagnated generations of particle i.

The second condition is that the mean distance among the current position of particle i and its recently previous n positions is smaller than a threshold value λ. This condition is described as follows:

$$\frac{\sum_{j=1}^{n} |pos_{ki} - pos_{k-j,i}|}{n} < \lambda, \tag{7}$$

where pos_{ki} is the position of particle i at the kth generation.

When both conditions are met, Michalewicz's non-uniform mutation operator is executed on particle i as follows (Michalewicz, 1996):

$$x_i^d = \begin{cases} x_i^d + delta(k, x_{\max}^d - x_i^d) & \text{if } rand() \leqslant 0.5 \\ x_i^d - delta(k, x_i^d - x_{\min}^d) & \text{otherwise} \end{cases}, \tag{8}$$

where k is the current generation number; $x_{\max}^d$ and $x_{\min}^d$ are the upper bound and lower bound of the dth dimension, respectively. Afterward $delta(k, u)$ generates a value among $[0, u]$ in the following manner

$$delta(k, u) = u(1 - r^{(1-k/MaxG)^b}), \tag{9}$$

where r is a random value uniformly distributed among $[0,1]$. $MaxG$ is the maximum generations allowed in the algorithm; b is a tuneable parameter conventionally set to 5 (Michalewicz, 1996). Michalewicz's non-uniform mutation operator enable particles to explore large area at the early stage while reducing its mutation amplitude with the evolution to protect the convergence.

Apparently, the mutation operator in this study is completely based on the individual level, which is beneficial to the adaptation and flexibility of each particle. In addition, the two conditions used to recognise the trap state can provide the mutation more accurate guidance.

6. Framework of SSG-PSO

Based on the exploration and exploitation components described previously, the SSG-PSO framework is given in Algorithm 3. The termination is controlled by both the consumed generations and function evaluations. Notably, all local search methods employed in this study, including BFGS, DFP, Pattern Search and Nelder–Mead simplex search, can be realised by calling the built-in functions of "fminunc", "fminsearch" and "patternsearch" provided by the Matlab software. After the call of these functions, the consumed function evaluations can be returned. Given that the local search can be costly in computation, it is only executed on the global best solution at the later evolutionary stage of PSO (indicated by $fitcount > \beta * MaxEF$ in the algorithm).

Algorithm 3: Framework of SSG-PSO

```
Initialize MaxG, maximum function evaluations MaxEF,
   learning dimension number m;
Initialize the cardinality of S2, swarm size ps
Initialize the position x_i and velocity v_i for each particle i;
Let pBest_i ← x_i and calculate the gBest;
Add each pBest_i to S1, set S2 = ∅;
sTag_i = 0;
Set k = 1, fitcount = ps;
While k < MaxG && fitcount < MaxEF
  k = k + 1;
  For i = 1 → ps
    Randomly select m dimensions to form the set M;
    Update the velocity of particle i according to (3)
    Update the position of particle i according to (2);
    fitcount = fitcount + 1;
  If f(pos_i) < f(pBest_i)
      If ∃j ∈ S2, f(pBest_i) < f(pos_j)
        pBest_i is used to replace the worst solution in S2;
      End if
      Update pBest_i with pos_i;
      sTag_i = 0;
    Else
      If ∃j ∈ S2, f(pos_i) < f(pos_j)
        pos_i is used to replace the worst solution in S2;
      End if
      sTag_i = sTag_i + 1;
    End if
    If f(pos_i) < f(gBest)
      If fitcount > β * MaxEF
        Execute the local search (BFGS, DFP, Pattern Search
  or Simplex Search) on pos_i;
        Let nCount be the function evaluations consumed by
  the Local Search;
        fitcount = fitcount + nCount;
      End if
      Update pos_i with gBest;
    End if
    If both conditions (6) and (7) are satisfied
      Execute the mutation operator on particle i according to
  (8);
    End if
  End for
End while
```

7. Experimental studies

7.1. Experimental setting

To comprehensively compare the performance of different PSO variants, 21 test benchmark functions of different types were used, including unimodal functions, multimodal functions, mis-scaled functions and rotated functions. The detailed information of the test functions are displayed in Table 1. **M** is the orthogonal matrix.

To evaluate the performance of SSG-PSO and its several variants with different local search techniques, SSG-PSO were compared with other popular PSO variants, which are listed below.

FIPS-PSO: Fully informed PSO (Mendes et al., 2004)
CPSO-H: Cooperative based PSO (Van den Bergh & Engelbrecht, 2004)
DMSPSO-LS: Dynamic multi-swarm particle swarm optimiser with local search (Liang & Suganthan, 2005)
CLPSO: Comprehensive learning PSO (Liang et al., 2006)
APSO: Adaptive particle swarm optimization (Zhan et al., 2009)
The parameter settings of the comparative algorithms are the same as that given in the corresponding references.

Different local search methods are combined into SSG-PSO. To comprehensively and experimentally compare the improvements brought by different local search techniques; different variants of SSG-PSO with different local search techniques are derived and studied in the experiments.

SSG-PSO: Superior solutions guided PSO with the individual level based mutation operator.
SSG-PSO-DFP: SSG-PSO with DFP method.
SSG-PSO-BFGS: SSG-PSO with BFGS method.
SSG-PSO-NM: SSG-PSO with Nelder–Mead simplex search.
SSG-PSO-PS: SSG-PSO with Pattern Search.

Parameter values used in the SSG-PSO variants are given below.

Cardinality of S_2 $|S_2| = 20$, $m = 2$, $t = 50$, $\lambda = 0.0001$, $\beta = 0.8$, $n = 5$, $b = 5$, $w = 1 - 0.5k/MaxG$, $c = 2$, $MaxG = 7500$, $MaxEF = 3e+5$, $ps = 40$.

In addition, when using "fminunc", "fminsearch" and "patternsearch" functions of Matlab to realise the local search methods, the termination conditions are set.

Maximum number of objective function evaluations: *MaxFunEvals* = 3000; Tolerance on function: *TolFun* = 1e−20; Tolerance on variable: *TolX* = 1e−20.

7.2. Comparative results

The computational results of each comparative algorithm on the benchmark functions are listed in Tables 2 and 3, where "Mean" is the average value of the objective function values obtained by running each comparative algorithm 30 times; "Std" is the related standard deviation value; "Suc" is the number of runs which generate satisfactory solutions for each function. For all the benchmark functions, the theoretically optimal objective function value is equal to zero. Therefore, for any benchmark function f_i, if $f_i(x^*) < 1e-5$, x^* is a satisfactory solution of f_i. "Rank" records the performance ranks of the comparative algorithms in dealing with each optimization function based on their obtained mean results ("Mean"). In addition, the comprehensive comparison results with respect to the ranks of different algorithms are given in Table 4. From the results displayed in Tables 2–4, we can get some observations.

For unimodal optimization functions, algorithm SSG-PSO-BFGS and SSG-PSO-DFP exhibit promising performance. Especially, SSG-PSO-BFGS is most efficient in solving each unimodal optimization function, except that SSG-PSO-PS obtains the best results for function f_4. In addition, DMSPSO-LS is also very competitive in dealing with function f_1, f_2, f_3 and f_4, demonstrating that local search techniques can significantly improve the performance of PSO when solving unimodal optimization problems, because the exploitation capability of PSO is significantly enhanced by local search techniques. For non-differential optimization function f_4, the PSO integrated with derivative-free Pattern Search (SSG-PSO-PS) demonstrates the best performance, indicating the potential of the combination of this derivative-free local search technique into PSO to solve discontinuous or non-differential optimization problems more efficiently. In addition, the gradient-based local search techniques are also useful to support PSO to tackle non-differential optimization functions (e.g., f_4 is completely non-differential and f_5 is not differential everywhere). The reason is that the gradient-based

Table 1
Benchmark optimization functions.

Type	Function formula	Name	Range
Unimodal	$f_1(x)=\sum_{i=1}^{n}x_i^2$	Sphere	[−500,500]
	$f_2(x)=\sum_{i=1}^{n}(100(x_i^2-x_{i+1})^2+(x_i-1)^2)$	Rosenbrock	[−2.048, 2.048]
	$f_3(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i}x_j^2)$	Schwefel 1.2	[−100,100]
	$f_4(x)=\max\{\lvert x_i\rvert,1\leqslant i\leqslant D\}$	Schwefel 2.21	[−10,10]
	$f_5(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	Schwefel 2.22	[−10,10]
Multimodal	$f_6(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	Rastrigin	[−5.12, 5.12]
	$f_7(x)=\sum_{i=1}^{n}(y_i^2-10\cos(2\pi y_i)+10), y_i=\begin{cases}x_i, if\lvert x_i\rvert<0.5\\0.5*round(2*x_i), otherwise\end{cases}$	Noncontinues Rastrigin	[−600, 600]
	$f_8(x)=-20\cdot\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})+20-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+e$	Ackley	[−32, 32]
	$f_9(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2+1-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	Griewank	[−600, 600]
Mis-scaled	$f_{10}(x)=\sum_{i=1}^{n}(100((a_ix_i)^2-(a_{i+1}x_{i+1}))^2+(a_ix_i-1)^2), a_i=100^{\frac{i-1}{n-1}}$	Scaled Rosenbrock 100	[−4.196,4.196]
	$f_{11}(x)=\sum_{i=1}^{n}((a_ix_i)^2-10\cos(2\pi(a_ix_i))+10), a_i=10^{\frac{i-1}{n-1}}$	Scaled Rastrigin 10	[−5.12,−5.12]
	$f_{12}(x)=\sum_{i=1}^{n}((a_ix_i)^2-10\cos(2\pi(a_ix_i))+10), a_i=1000^{\frac{i-1}{n-1}}$	Scaled Rastrigin 1000	[−5.12,−5.12]
Rotated	$f_{13}(x)=\sum_{i=1}^{n}z_i^2, \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Sphere	[−500,500]
	$f_{14}(x)=\sum_{i=1}^{n}(100(z_i^2-x_{i+1})^2+(z_i-1)^2), \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Rosenbrock	[−2.048, 2.048]
	$f_{15}(x)=\max\{\lvert z_i\rvert,1\leqslant i\leqslant D\}, \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Schwefel 2.21	[−10,10]
	$f_{16}(x)=\sum_{i=1}^{n}(z_i^2-10\cos(2\pi z_i)+10), \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Rastrigin	[−5.12, 5.12]
	$f_{17}(x)=-20\cdot\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}z_i^2})+20-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi z_i))+e, \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Ackley	[−32, 32]
	$f_{18}(x)=\frac{1}{4000}\sum_{i=1}^{n}z_i^2+1-\prod_{i=1}^{n}\cos(\frac{z_i}{\sqrt{i}}), \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Griewank	[−600, 600]
	$f_{19}(x)=\sum_{i=1}^{n}(20^{\frac{i-1}{n-1}}z_i)^2, \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Ellipse	[−100, 100]
	$f_{20}(x)=(1000x_1)^2+\sum_{i=2}^{n}z_i^2, \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated Tablet	[−100, 100]
	$f_{21}(x)=\sum_{i=1}^{n}z_i^{2+a_i}, a_i=10^{\frac{i-1}{D-1}}\ \mathbf{z}=\mathbf{M}\cdot\mathbf{x}$	Rotated diff pow	[−100, 100]

Table 2
Computational results of f_1-f_{15}. The best results of each benchmark function are shown in boldface.

Algorithm	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank
Functions	*f_1 Sphere*				*f_2 Rosenbrock*				*f_3 Schwefel 1.2*			
FIPS-PSO	**0.00E+00**	**0.00E+00**	**30**	**1**	2.52E+01	9.08E−01	0	9	2.08E+02	8.98E+01	0	9
CPSO-H	2.49E−15	1.55E−15	30	10	2.77E+01	2.86E+01	0	10	2.79E+03	5.98E+03	0	10
CLPSO	**0.00E+00**	**0.00E+00**	**30**	**1**	2.34E+01	3.79E+00	0	7	1.34E+02	2.44E+02	0	8
APSO	**0.00E+00**	**0.00E+00**	**30**	**1**	2.38E+01	7.05E+01	0	8	9.91E−03	5.03E−02	0	4
DMSPSO-LS	**0.00E+00**	**0.00E+00**	**30**	**1**	8.99E−11	3.54E−11	30	2	9.59E−10	5.33E−10	0	3
SSG-PSO	**0.00E+00**	**0.00E+00**	**30**	**1**	2.12E+01	1.84E+00	0	6	1.24E+01	1.35E+01	0	6
SSG-PSO-DFP	**0.00E+00**	**0.00E+00**	**30**	**1**	1.84E−01	4.35E−01	0	3	1.30E−03	2.12E−03	0	2
SSG-PSO-BFGS	**0.00E+00**	**0.00E+00**	**30**	**1**	**5.73E−11**	**8.08E−12**	**30**	**1**	**2.57E−14**	**4.58E−14**	**30**	**1**
SSG-PSO-NM	**0.00E+00**	**0.00E+00**	**30**	**1**	8.34E+00	2.83E+00	0	5	2.46E−02	4.14E−02	0	5
SSG-PSO-PS	**0.00E+00**	**0.00E+00**	**30**	**1**	6.90E+00	1.25E+00	0	4	4.16E+01	2.31E+00	0	7
Functions	*f_4 Schwefel 2.21*				*f_5 Schwefel 2.22*				*f_6 Rastrigin*			
FIPS-PSO	6.25E−02	4.34E−03	0	9	1.64E−09	5.59E−10	30	9	6.39E+01	1.12E+01	0	10
CPSO-H	5.14E−03	4.49E−03	0	5	3.92E−08	4.25E−09	30	10	3.32E−02	1.82E−01	5	7
CLPSO	6.61E−03	4.21E−03	0	6	7.52E−20	5.69E−19	30	7	**0.00E+00**	**0.00E+00**	**30**	**1**
APSO	2.24E−01	6.85E−01	0	10	4.25E−23	7.44E−21	30	5	4.52E+00	1.35E+00	0	8
DMSPSO-LS	4.67E−05	1.75E−05	10	3	1.09E−18	1.04E−18	30	8	1.32E+01	1.93E+00	0	9
SSG-PSO	2.77E−02	1.47E−02	0	8	2.75E−25	1.96E−25	30	2	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-DFP	1.60E−04	2.92E−06	2	4	9.65E−25	6.08E−25	30	3	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-BFGS	3.77E−06	1.63E−06	30	2	**1.04E−26**	**7.54E−26**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-NM	2.12E−02	1.11E−02	0	7	1.93E−24	1.21E−24	30	4	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-PS	**1.23E−15**	**3.25E-16**	**30**	**1**	9.33E−22	1.23E−22	30	6	**0.00E+00**	**0.00E+00**	**30**	**1**
Functions	*f_7 Noncontinues Rastrigin*				*f_8 Ackley*				*f_9 Griewank*			
FIPS-PSO	5.44E+01	2.63E+01	0	10	1.39E−08	2.98E−09	30	8	2.72E−07	1.18E−06	30	8
CPSO-H	2.56E−09	7.43E−09	30	7	2.44E−05	1.35E−05	1	9	1.20E−01	2.18E−01	4	10
CLPSO	**0.00E+00**	**0.00E+00**	**30**	**1**	7.77E−14	1.49E−18	30	7	**0.00E+00**	**0.00E+00**	**30**	**1**
APSO	3.21E+00	6.32E+00	0	8	6.34E−02	1.43E+00	0	10	1.42E−02	7.24E−02	0	9
DMSPSO-LS	3.41E+01	5.02E+00	0	9	7.81E−15	2.80E−15	30	5	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO	**0.00E+00**	**0.00E+00**	**30**	**1**	7.25E−15	1.74E−16	30	4	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-DFP	**0.00E+00**	**0.00E+00**	**30**	**1**	5.68E−15	1.78E−15	30	2	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-BFGS	**0.00E+00**	**0.00E+00**	**30**	**1**	**4.97E−15**	**1.73E−15**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-NM	**0.00E+00**	**0.00E+00**	**30**	**1**	6.85E−15	3.23E−15	30	3	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-PS	**0.00E+00**	**0.00E+00**	**30**	**1**	1.25E−14	6.32E−15	30	6	**0.00E+00**	**0.00E+00**	**30**	**1**
Functions	*f_{10} Scaled Rosenbrock 100*				*f_{11} Scaled Rastrigin 10*				*f_{12} Scaled Rastrigin 100*			
FIPS-PSO	7.37E+04	3.14E+05	0	8	7.37E+04	9.23E+00	0	9	4.59E+01	2.38E+01	0	10

Table 2 (*continued*)

Algorithm	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank
CPSO-H	3.71E+06	4.38E+06	0	9	1.23E+07	1.86E−07	0	10	5.65E−05	2.44E-04	12	7
CLPSO	4.94E+01	4.32E+01	0	7	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
APSO	4.06E+06	5.42E+06	0	10	1.98E+00	2.44E+01	0	7	1.49E+01	5.24E+01	0	8
DMSPSO-LS	2.56E+01	1.02E+01	0	3	2.17E+01	6.70E+00	0	8	3.06E+01	6.41E+00	0	9
SSG-PSO	3.59E+01	3.22E+01	0	6	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-DFP	3.19E+01	3.18E+01	0	4	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-BFGS	2.21E+01	3.07E+01	0	2	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-NM	3.43E+01	3.03E+01	0	5	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
SSG-PSO-PS	**1.81E+00**	**1.05E+00**	**0**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**	**0.00E+00**	**0.00E+00**	**30**	**1**
Functions	*f_{13} Rotated Sphere*				*f_{14} Rotated Rosenbrock*				*f_{15} Rotated Schwefel 2.21*			
FIPS-PSO	7.54E−13	3.26E−13	30	9	2.89E+01	4.15E+00	0	8	1.36E−04	4.89E−05	0	4
CPSO-H	8.10E−08	1.02E−07	30	10	1.62E+02	3.78E+02	0	10	5.43E+01	7.52 + 00	0	10
CLPSO	4.21E−17	6.18E−18	30	8	2.64E+01	1.21e+00	0	7	1.51E−02	4.64E−03	0	8
APSO	3.24E−20	5.45E−19	30	7	7.83E+01	8.24E+01	0	9	8.05E−01	1.24E−01	0	9
DMSPSO-LS	**2.59E−30**	**1.87E−30**	**30**	**1**	3.98E−03	2.35E−03	0	3	1.18E−04	1.09E−04	0	3
SSG-PSO	5.28E−22	8.71E−22	30	6	2.53E+01	6.91E−01	0	6	7.25E−04	3.02E−04	0	7
SSG-PSO-DFP	1.91E−23	3.04E−23	30	5	1.06E−05	7.88E−04	18	2	**5.38E−06**	**8.59E−06**	**30**	**1**
SSG-PSO-BFGS	2.29E−27	3.74E−27	30	2	**3.98E−10**	**1.22E−10**	**30**	**1**	8.15E−06	7.15E−06	30	2
SSG-PSO-NM	5.32E−24	4.11E−24	30	4	2.40E+01	1.36E+01	0	4	4.94E−04	1.96E-04	0	6
SSG-PSO-PS	1.97E-24	8.65E-25	30	3	2.50E+01	1.41E+01	0	5	4.40E-04	2.31E−04	0	5

Table 3
Computational results of $f_{16} - f_{21}$. The best results of each benchmark function are shown in boldface.

Algorithm	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank	Mean	Std	Suc	Rank
Functions	*f_{16} Rotated Rastrigin*				*f_{17} Rotated Ackley*				*f_{18} Rotated Griewank*			
FIPS-PSO	1.75E+02	8.79E+00	0	9	2.24E−08	5.60E−09	30	8	1.14E-03	3.00E−03	8	7
CPSO-H	3.77E+02	1.10E+02	0	10	1.76E+01	3.96E+00	0	10	1.66E+00	2.10E−01	0	10
CLPSO	1.08E+02	1.36E+01	0	8	2.76E−03	3.25E−03	0	9	2.66E−03	2.12E−03	0	8
APSO	1.02E+02	1.24E+03	0	7	3.62E−10	9.94E−10	30	7	1.72E−02	2.41E−01	0	9
DMSPSO-LS	3.10E+01	4.54E+00	0	2	2.48E−14	5.84E−15	30	3	7.39E−04	2.33E−03	0	6
SSG-PSO	4.65E+01	1.12E+01	0	5	5.86E−14	1.55E−13	30	4	1.02E−05	4.86E−05	13	5
SSG-PSO-DFP	5.08E+01	1.16E+01	0	6	4.79E−15	1.73E−15	30	2	**1.11E−16**	**3.02E−16**	**30**	**1**
SSG-PSO-BFGS	4.10E+01	1.43E+01	0	3	**4.59E−15**	**2.64E−15**	**30**	**1**	1.47E−16	3.03E−16	30	2
SSG-PSO-NM	**5.38E+00**	**1.05E+01**	**0**	**1**	7.12E−14	1.67E−13	30	6	1.10E−08	2.70E−08	30	3
SSG-PSO-PS	4.44E+01	6.43E+01	0	4	6.16E−14	1.95E−13	30	5	1.52E−06	2.84E−06	30	4
Functions	*f_{19} Rotated Ellipse*				*f_{20} Rotated Tablet*				*f_{21} Rotated diff pow*			
FIPS-PSO	1.51E+03	7.14E+02	0	8	8.45E+02	2.14E+02	0	9	4.54E+10	9.01E+10	0	10
CPSO-H	7.63E+03	6.69E+03	0	10	1.65E+05	4.61E+04	0	10	1.66E+10	4.71E+10	0	9
CLPSO	4.88E+03	1.38E+03	0	9	3.35E+02	1.34E+02	0	7	3.74E+07	3.45E+07	0	8
APSO	1.25E+03	2.12E+04	0	7	7.41E+02	8.44E+02	0	8	2.91E+07	4.22E+08	0	7
DMSPSO-LS	1.27E−09	1.13E−09	30	2	1.37E−07	1.47E−07	30	4	1.28E−08	6.28E−09	30	2
SSG-PSO	7.72E+02	8.86E+02	0	6	3.02E+02	1.10E+02	0	6	5.42E+06	7.17E+06	0	6
SSG-PSO-DFP	2.54E−07	1.07E−06	30	3	1.30E−11	4.44E−11	30	2	1.45E−06	9.86E−07	30	3
SSG-PSO-BFGS	**2.44E−16**	**4.38E−16**	**30**	**1**	**2.23E−12**	**2.40E−12**	**30**	**1**	**8.87E−10**	**5.79E−10**	**30**	**1**
SSG-PSO-NM	6.17E−01	8.93E−01	0	4	4.91E−11	2.10E−10	30	3	7.73E+03	1.44E+04	0	4
SSG-PSO-PS	5.13E+01	3.24E+01	0	5	2.14E−02	6.54E−02	0	5	3.26E+04	2.16E+04	0	5

Table 4
Comprehensive comparison results of different algorithms.

	FIPS-PSO	CPSO-H	CLPSO	APSO	DMSPSO-LS	PSG-PSO	SSG-PSO-DFP	SSG-PSO-BFGS	SSG-PSO-NM	SSG-PSO-PS
Mean rank	8.1905	9.1905	5.7143	7.5238	4.1429	4.2381	2.3333	1.3333	3.3333	3.4286
Overall rank	9	10	7	8	5	6	2	1	3	4

Table 5
The candidate values of the parameters for sensitivity analysis.

Parameters	Candidate values
$\|S_2\|$	{15, 16, 17, ..., 23, 24, 25}
m	{1, 2, 3, ..., 18, 19, 20}
β	{0.1, 0.2, 0.3,..., 0.7, 0.8, 0.9}
λ	{0.000001, 0.00001, 0.0001, 0.0005, 0.001, 0.005, 0.1, 0.5, 1, 5}

local search techniques (i.e., BFGS and DFP) can utilise numerical methods to calculate the approximate gradient. As a result, they may obtain the right modification direction of each variable of the optimization function.

For multi-modal optimization functions, algorithms CLPSO, SSG-PSO and its local search enhanced variants all showed promising efficiency because CLPSO and SSG-PSO have high exploration capability, which is crucial for solving multi-modal optimization problems. Notably, DMSPSO-LS is not efficient in dealing with

function f_6, f_7 and f_9 although it is combined with a local search technique. This result reveals that combining the local search techniques with a PSO variant which has strong exploration capability would be more useful.

For mis-scaled optimization functions, CLPSO, SSG-PSO and its local search enhanced variants are the most efficient in solving functions f_{11} and f_{12}, because these two functions are also multi-modal. Interestingly, SSG-PSO-PS outperforms SSG-PSO-BFGS and DMSPSO-LS in solving function f_{10}, although this function is differential.

For rotated optimization problems, PSO variants with local search techniques generally perform better than that without local search techniques because variables in rotated optimization functions are highly related, which makes it hard for an EA to find the proper modification directions of the variables. Therefore, traditional PSO variants face two difficulties in solving rotated optimization functions. The first one is that it is difficult for them to step toward a high-quality solution area, thereby slowing better solution update process. The second one is when the local best and global best solutions are not updated for a long while, the whole swarm can get trapped easily. By contrast, in a certain degree, the local search techniques can provide more accurate and effective guidance of search directions for the particles. Given that the selected rotated functions are different, PSO variants with gradient-based local search techniques are usually more efficient than that with derivative-free local search techniques, although SSG-PSO-NM is the best for function f_{16}. Notably SSG-PSO-BFGS showed noticeably better performance in solving function f_{19}, f_{20} and f_{21}.

Based on the overall performance comparison results shown in Table 4, (1) SSG-PSO-BFGS has the best overall performance in solving the benchmark optimization functions; (2) PSO variants combined with local search techniques are superior to PSO variants without local search techniques; (3) Compared with other peer PSO variants without local search techniques, SSG-PSO shows competitive performance.

SSG-PSO with BFGS and DFP are the two algorithms with the highest overall performance in solving benchmark optimization problems. Notably, the integration of such gradient-based search techniques increases the computational burden because the Hessian matrix needs to be numerically approximated. However, these local search techniques can converge to better solutions (local optima) more efficiently with more accurate information guiding the search behaviour. Therefore, the computation consumption caused by the local search technique is worthwhile. The function evaluations consumed by the used local search techniques are recorded and considered into the calculation of the total consumed function evaluations, when local search enhanced SSG-PSO variants are applied to the optimization problems. Therefore, comparisons between our results and those obtained by other peer algorithms are fair.

Local search techniques may bring the risk of premature convergence (i.e., get trapped in local optima). To avoid such risks, the algorithm framework SSG-PSO to be combined with the local search techniques employs two strategies, namely, superior solution guided comprehensive learning strategy and individual level based mutation strategy, which enable SSG-PSO to have a strong exploration capability.

The following findings were obtained according to the computational results and comparative analyses. First, to guarantee effectiveness, the PSO to be combined with local search techniques should have strong exploration capability, which is crucial for solving multi-modal optimization problems. Second, compared

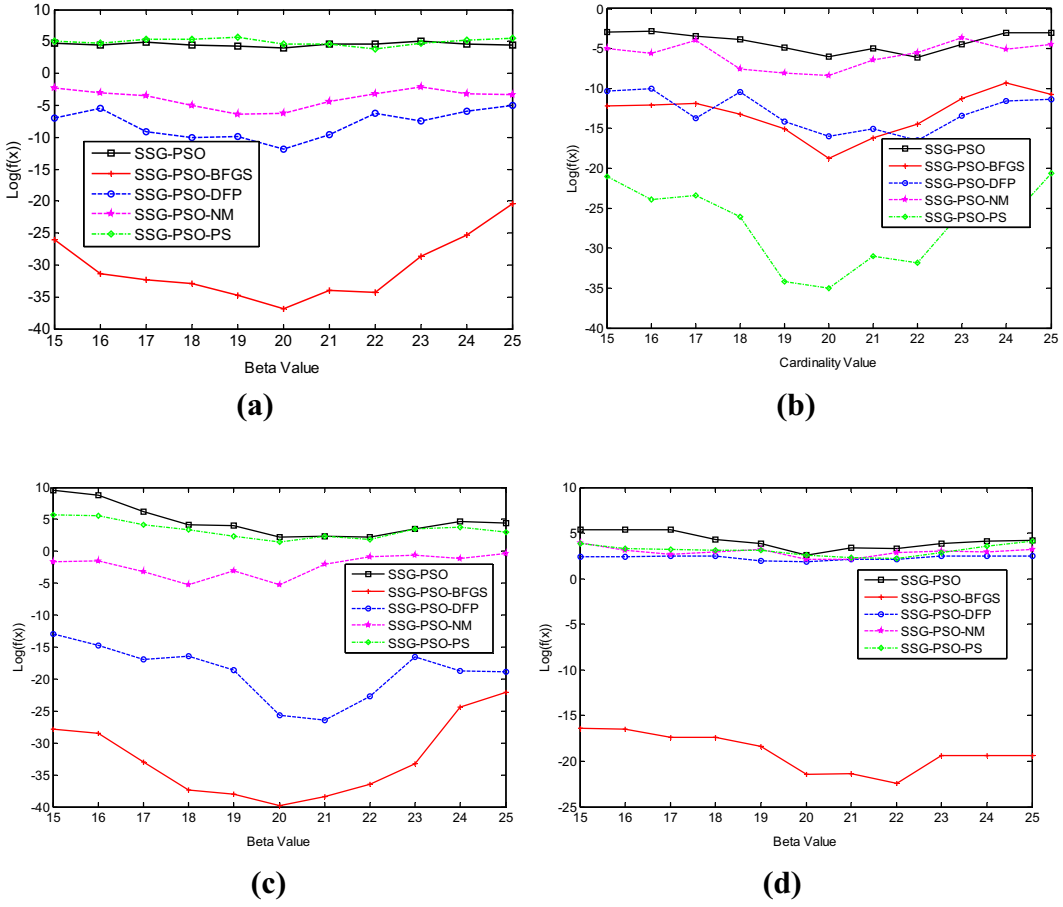

Fig. 1. Sensitivity of the cardinality of comprehensive learning solution set, (a) f_3; (b) f_4; (c) f_{14} and (d) f_{19}.

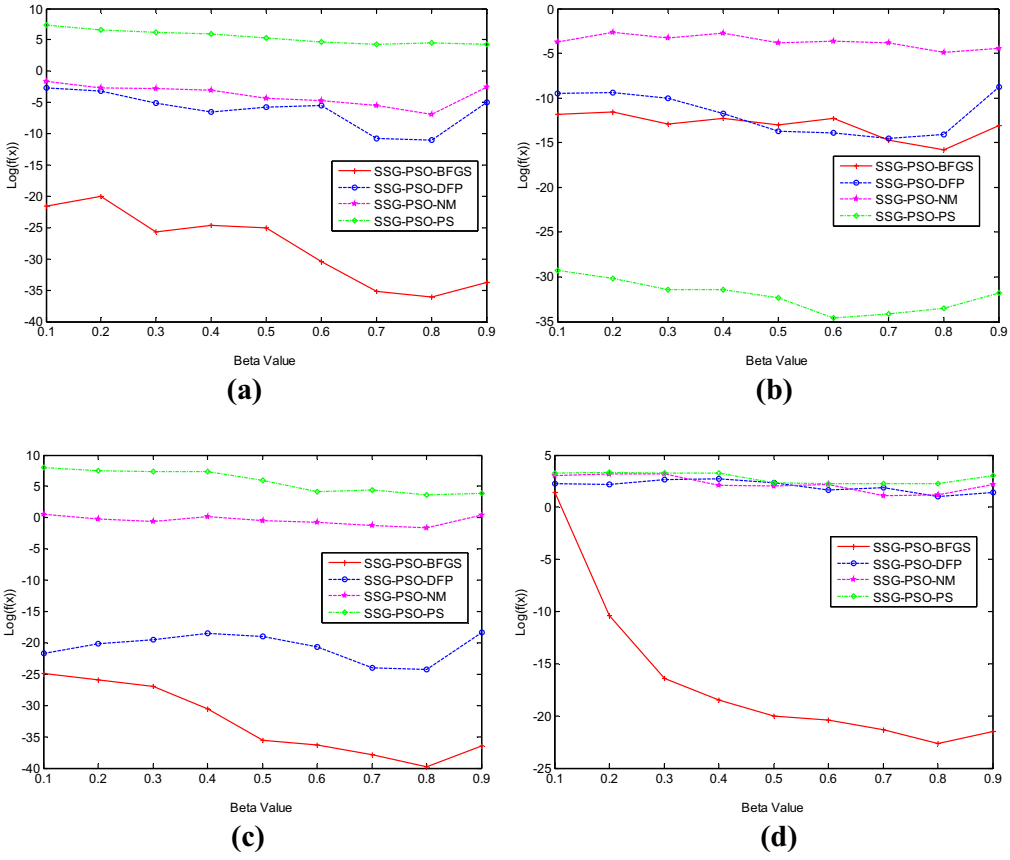

Fig. 2. Sensitivity of beta, (a) f_3; (b) f_4; (c) f_{14} and (d) f_{19}.

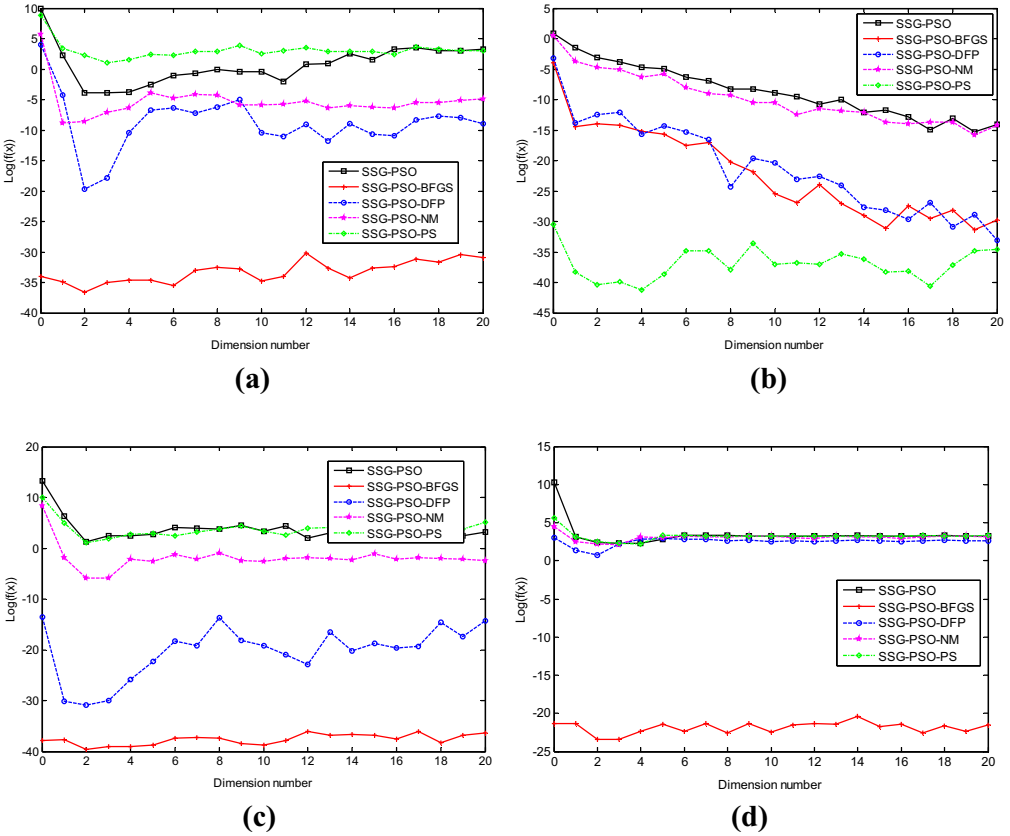

Fig. 3. Sensitivity of dimension number for comprehensive learning, (a) f_3; (b) f_4; (c) f_{14} and (d) f_{19}.

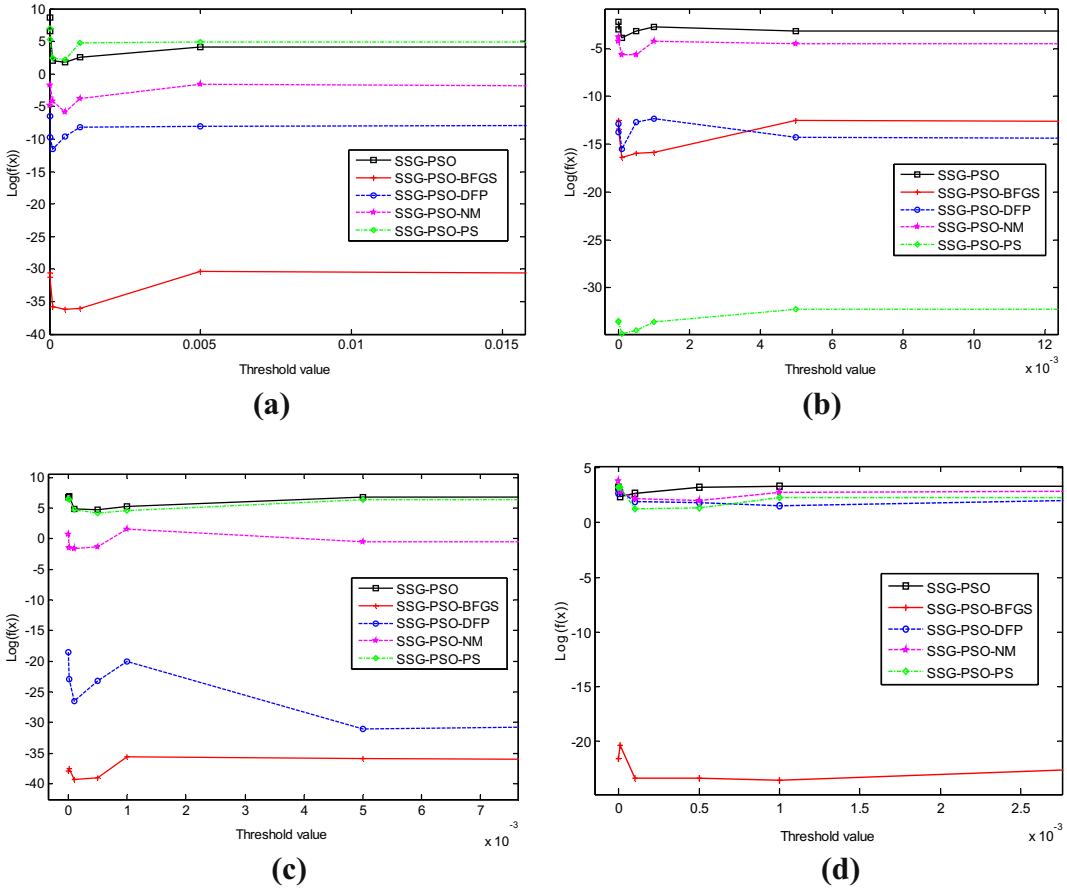

Fig. 4. Sensitivity of the mutation threshold value, (a) f_3; (b) f_4; (c) f_{14} and (d) f_{19}.

with the DFP method, Pattern Search, and Nelder–Mead simplex search, the BFGS method is the most useful local search technique which can be incorporated into PSO. However, different local search techniques may have special advantages in solving different optimization problems. For example, Nelder–Mead simplex search is more efficient for function f_{16}, whereas Pattern Search is a better choice for function f_4. Third, for differential optimization problems, gradient-based local search techniques are more efficient. For non-differential optimization problems, given that gradient-based local search techniques such as BFGS use numerical methods to calculate the gradient, they may also be suited to non-differential optimization problems. Fourth, the performance of SSG-PSO is better than other competitive PSO variants, such as FIPS-PSO, CPSO-H, CLPSO and APSO. This finding indicates that the adopted superior solution guided leaning strategy and the individual level based mutation operator are effective in improving the performance of traditional PSO.

7.3. Parameter sensitivity analysis

The performance of a meta-heuristic is greatly influenced by its parameter values. Therefore, the impact of different parameter values on the performance of the algorithms must be experimentally analysed. In this study, the parameter values were configured by traditional trial-and-error approach. The appropriate parameter values used to solve the benchmark optimization functions are given in Section 7.1. Four parameters with higher impact on the performance of the algorithms are analysed. The candidate values of the parameters prepared for the sensitivity analysis are given in Table 5. To show the parameter sensitivity, each algorithm with different parameter values is applied to solve four representative optimization functions, including f_3, f_4, f_{14} and f_{19}. Notably, when one parameter is analysed, the other parameters are fixed to the proper values provided in Section 7.1.

The results are graphically demonstrated in Figs. 1–4, from which the value settings of parameter $|S_2|$, m, β and λ generally have noticeable impact on the performance of each algorithm. Given that the proposed algorithms share the same framework, the influences of the parameters on the algorithms are consistent. However, two things must also be considered. First, when an algorithm is applied to different problems, the optimal parameter values of this algorithm may still be slightly different. For example, the appropriate cardinality value of the superior solution set is set to 20; however, the exact optimal cardinality value for SSG-PSO-BFGS solving f_{14} is 22. Second, the best parameter values for different algorithms are not always exactly same. For instance, when solving f_4, the best value of β for SSG-PSO-PS is 0.6, whereas the most appropriate value of β for SSG-PSO-BFGS is 0.8. When rational parameter values for the algorithms are set, the parameter values beneficial for major algorithms will be selected to guarantee fair comparison.

8. Conclusions

Exploration and exploitation is a pair of contradicting concepts which serve key functions in the design of efficient evolutionary algorithms. This study presented a superior solution guided particle swarm (SSG-PSO) with a novel individual level based mutation strategy. Different gradient-based and derivative-free local search techniques are incorporated into SSG-PSO, thereby obtaining several local search enhanced SSG-PSO variants. In SSG-PSO, a collection of superior solutions is maintained and the particles

comprehensively learn from the superior solutions. To further maintain the proper diversity of the swarm, the individual level based mutation strategy employs Michalewicz's non-uniform mutation operator, which will be triggered when a particle gets trapped with respect to its fitness and position states. The local search techniques combined into SSG-PSO include the BFGS and DFP Quasi-Newton methods, Pattern Search and Nelder–Mead simplex search.

The performance of SSG-PSO variants were extensively evaluated on a suite of 21 bound-constrained numerical optimization problems. Experimental results showed that SSG-PSO and its local search enhanced variants outperformed competitive PSO variants such as FIPS-PSO, CPSO-H, CLPSO and APSO on various benchmark unconstrained optimization problems.

The overall performance of SSG-PSO with BFGS method is best in solving optimization functions. However, SSG-PSO with Pattern Search and Nelder–Mead simplex search can be particularly efficient in specific optimization problems. Local search techniques can strengthen the performance of PSO. However, to gain a balanced exploration and exploitation capability, the PSO variant to be integrated with local search techniques should have promising exploration capability. Moreover, although the BFGS method is gradient-based, it can be applied to non-differential optimization problems because it uses numerical methods to calculate the required gradient in real applications.

The proposed local search enhanced SSG-PSO algorithms demonstrated powerful performance in solving a suit of benchmark unconstrained optimization problems with different characteristics. Unconstrained optimization problems widely exist in real-world applications, and are expected to have significant impact on various real-world unconstrained optimization problems, such as the antenna array design problem and the spacecraft trajectory optimization problem.

To solve differential unconstrained optimization problems, the combination of BFGS and SSG-PSO is generally a better choice. By contrast, to address non-differential optimization problems, trying to combine SSG-PSO with different local search techniques may be a good idea. However, the local search techniques employed in this study are specific for unconstrained optimization problems; thus, the proposed algorithms may be not suited to constrained optimization problems.

Meta-heuristics such as Ant Colony Optimization (ACO) and Harmony Search can have strong exploitation capability. However, in this study, the local search techniques combined into SSG-PSO are all classical mathematical programming methods. As a result, combining PSO with other meta-heuristics functioned as local search techniques can be interesting and meaningful. In addition, the local search techniques employed are only suited to unconstrained optimization problems. Therefore, to deal with constrained optimization problems more efficiently, it is useful to investigate the combination of PSO and some local search techniques oriented to constrained optimization problems, such as Interior-Point method and Sequential Quadratic Programming method.

Acknowledgements

This work was supported by the National Nature Science Foundation of China under Grant Nos. 71271213, 41001220, and 51178193, and China Scholarship Council under Grant Nos. 201206110082 and 201208430109.

References

Andrews, P. S. (2006). An investigation into mutation operators for particle swarm optimization. In *Proceeding IEEE congress on evolutionary computation* (pp. 1044–1051).

Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. *In Evolutionary Programming VII*, 601–610.

Banks, A., Vincent, J., & Anyakoha, C. (2007). A review of particle swarm optimization. Part I: Background and development. *Natural Computing, 6*, 467–484.

Banks, A., Vincent, J., & Anyakoha, C. (2008). A review of particle swarm optimization. Part II: Hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications. *Natural Computing, 7*, 109–124.

Boeringer, D. W., & Werner, D. H. (2004). Particle swarm optimization versus genetic algorithms for phased array synthesis. *IEEE Transactions on Antennas and Propagation, 52*, 771–779.

Bräuninger, J. (1981). A variable metric algorithm for unconstrained minimization without evaluation of derivatives. *Numerische Mathematik, 36*, 359–373.

Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & Operations Research, 33*, 859–871.

Chen, C. Y., Chang, K. C., & Ho, S. H. (2011). Improved framework for particle swarm optimization: Swarm intelligence with diversity-guided random walking. *Expert Systems with Applications, 38*, 12214–12220.

Chen, Y. P., Peng, W. C., & Jian, M. C. (2007). Particle swarm optimization with recombination and dynamic linkage discovery. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 37*, 1460–1470.

Chen, W. N., Zhang, J., Lin, Y., Chen, N., Zhan, Z. H., Chung, H. S. H., et al. (2013). Particle Swarm Optimization with an Aging Leader and Challengers. *IEEE Transactions on Evolutionary Computation, 17*, 241–258.

Costa, W. E., Goldbarg, M. C., & Goldbarg, E. G. (2012). Hybridizing VNS and path-relinking on a particle swarm framework to minimize total flow time. *Expert Systems with Applications, 39*, 13118–13126.

Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation, 15*, 4–31.

del Valle, Y., Venayagamoorthy, G. K., Mohagheghi, S., Hernandez, J. C., & Harley, R. G. (2008). Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Transactions on Evolutionary Computation, 12*, 171–195.

Eberhart, R., & Kennedy, J., (1995). A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science* (pp. 39–43).

Eberhart, R., Shi, Y., & Kennedy, J. (2001). *Swarm intelligence*. Morgan Kaufmann.

Fan, S.-K. S., & Zahara, E. (2007). A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research, 181*, 527–548.

Gao, H., & Xu, W. (2011). Particle swarm algorithm with hybrid mutation strategy. *Applied Soft Computing, 11*, 5129–5142.

Gholizadeh, S. (2013). Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization. *Computers & Structures, 125*, 86–99.

Gholizadeh, S., & Moghadas, R. K. (2014). Performance-based optimum design of steel frames by an improved quantum particle swarm optimization. *Advances in Structural Engineering, 17*, 143–156.

Gholizadeh, S., Torkzadeh, P., & Jabarzadeh, S. (2013). Optimum shape design of double-layer grids by quantum behaved particle swarm optimization and neural networks. *International Journal of Optimization in Civil Engineering, 3*, 85–98.

Hu, X., & Eberhart, R. (2002). Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation* (Vol. 2, pp. 1677–1681).

Hu, M., Wu, T., & Weir, J. D. (2012). An intelligent augmentation of particle swarm optimization with multiple adaptive methods. *Information Sciences, 213*, 68–83.

Huang, H., Qin, H., Hao, Z., & Lim, A. (2012). Example-based learning particle swarm optimization for continuous optimization. *Information Sciences, 182*, 125–138.

Ismail, A., & Engelbrecht, A. (2012). The self-adaptive comprehensive learning particle swarm optimizer. *Swarm Intelligence*, 156–167.

Juang, C. F. (2004). A hybrid of genetic algorithm and particle swarm optimization for recurrent network design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 34*, 997–1006.

Kaveh, A., & Talatahari, S. (2009a). A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research, 65*, 1558–1568.

Kaveh, A., & Talatahari, S. (2009b). Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers & Structures, 87*, 267–283.

Kennedy, J. (1997). The particle swarm: social adaptation of knowledge. In *IEEE International Conference on Evolutionary Computation* (pp. 303–308).

Kennedy, J. (1999). Small worlds and mega-minds: Effects of neighborhood topology on particle swarm performance. In *Proceedings of the 1999 congress on evolutionary computation* (Vol. 3).

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of IEEE international conference on neural networks* (Vol. 4, pp. 1942–1948).

Kennedy, J., & Mendes, R. (2002). Population structure and particle swarm performance. In *Proceedings of the 2002 congress on evolutionary computation* (Vol. 2, pp. 1671–1676).

Krohling, R. A. (2005). Gaussian particle swarm with jumps. In *Evolutionary computation, 2005*. The 2005 IEEE Congress on (Vol. 2, pp. 1226–1231).

Li, X. (2010). Niching without niching parameters: Particle swarm optimization using a ring topology. *IEEE Transactions on Evolutionary Computation, 14*, 150–169.

Li, C., Yang, S., & Nguyen, T. T. (2012). A self-learning particle swarm optimizer for global optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 42*, 627–646.

Liang, J., & Suganthan, P. (2005a). Dynamic multi-swarm particle swarm optimizer. In *Proceedings of the 2005 IEEE swarm intelligence symposium* (pp. 124–129).

Liang, J. J., Qin, A., Suganthan, P. N., & Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation, 10*, 281–295.

Liang, J. J., & Suganthan, P. N. (2005b). Dynamic multi-swarm particle swarm optimizer with local search. *In The 2005 IEEE Congress on Evolutionary Computation, 1*, 522–528.

Maruta, I., Kim, T. H., Song, D., & Sugie, T. (2013). Synthesis of fixed-structure robust controllers using a constrained particle swarm optimizer with cyclic neighborhood topology. *Expert Systems with Applications, 40*, 3595–3605.

Mendes, R., Kennedy, J., & Neves, J. (2004). The fully informed particle swarm: simpler, maybe better. *IEEE Transactions on Evolutionary Computation, 8*, 204–210.

Michalewicz, Z. (1996). *Genetic algorithms+ data structures = evolution programs*. Springer.

Nocedal, J., & Wright, S. J. (2000). *Numerical optimization*. Springer (2nd ed.).

Noel, M. M. (2012). A new gradient based particle swarm optimization algorithm for accurate computation of global minimum. *Applied Soft Computing, 12*, 353–359.

Omran, M. G. H., Engelbrecht, A. P., & Salman, A. (2007). Differential evolution based particle swarm optimization. In *IEEE Swarm Intelligence Symposium* (pp. 112–119).

Parsopoulos, K., & Vrahatis, M. (2007). Parameter selection and adaptation in unified particle swarm optimization. *Mathematical and Computer Modelling, 46*, 198–213.

Pehlivanoglu, Y. (2013). A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks. *IEEE Transactions on Evolutionary Computation, 17*, 436–452.

Plevris, V., & Papadrakakis, M. (2011). A hybrid particle swarm-gradient algorithm for global structural optimization. *Computer-Aided Civil and Infrastructure Engineering, 26*, 48–68.

Poli, R., Di Chio, C., & Langdon, W. B. (2005). Exploring extended particle swarms: A genetic programming approach. In *Proceedings of the 2005 conference on genetic and evolutionary computation* (pp. 169–176). ACM.

Poli, R., Kennedy, J., & Blackwell, T. (2007). Particle swarm optimization. *Swarm Intelligence, 1*, 33–57.

Qu, B., Liang, J., & Suganthan, P. (2012). Niching particle swarm optimization with local search for multi-modal optimization. *Information Sciences, 197*, 131–143.

Qu, B. Y., Suganthan, P. N., & Das, S. (2013). A distance-based locally informed particle swarm model for multimodal optimization. *IEEE Transactions on Evolutionary Computation, 17*, 387–402.

Rana, S., Jasola, S., & Kumar, R. (2011). A review on particle swarm optimization algorithms and their applications to data clustering. *Artificial Intelligence Review, 35*, 211–222.

Ratnaweera, A., Halgamuge, S. K., & Watson, H. C. (2004). Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. *IEEE Transactions on Evolutionary Computation, 8*, 240–255.

Shi, Y., & Eberhart, R. (1998a). A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE international conference on evolutionary computation* (pp. 69–73).

Shi, Y., & Eberhart, R. C. (1999). Empirical study of particle swarm optimization. In *Proceedings of the 1999 congress on evolutionary computation* (Vol. 3).

Shi, Y., & Eberhart, R. C. (2001). Fuzzy adaptive particle swarm optimization. In *Proceedings of the 2001 congress on evolutionary computation* (Vol. 1, pp. 101–106).

Shi, Y., & Eberhart, R. (1998b). *Parameter selection in particle swarm optimization. In Evolutionary Programming VII*. Springer, pp. 591–600.

Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. In *Proceedings of the 1999 congress on evolutionary computation* (Vol. 3).

Sun, W., & Yuan, Y.-X. (2006). *Optimization theory and methods: nonlinear programming*. Springer (Vol. 98).

Van den Bergh, F., & Engelbrecht, A. P. (2004). A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*, 225–239.

Wang, H., Liu, Y., Wu, Z., Sun, H., Zeng, S., & Kang, L. (2008). An improved particle swarm optimization with adaptive jumps. *In IEEE Congress on Evolutionary Computation*, 392–397.

Wang, H., Wang, W. J., & Wu, Z. J. (2013). Particle swarm optimization with adaptive mutation for multimodal optimization. *Applied Mathematics and Computation, 221*, 296–305.

Wei, C., He, Z., Zhang, Y., & Pei, W. (2002). Swarm directions embedded in fast evolutionary programming. In *Proceedings of the 2002 congress on evolutionary computation* (Vol. 2, pp. 1278–1283).

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

Wu, G., Ma, M., Zhu, J., & Qiu, D. (2012). Multi-satellite observation integrated scheduling method oriented to emergency tasks and common tasks. *Journal of Systems Engineering and Electronics, 23*, 723–733.

Wu, G., Pedrycz, W., Li, H., Qiu, D., Ma, M., & Liu, J. (2013). Complexity reduction in the use of evolutionary algorithms to function optimization: a variable reduction strategy. *The Scientific World Journal, 2013*, 1–8.

Wu, G., Pedrycz, W., Ma, M., Qiu, D., Li, H., & Liu, J. (2014). A particle swarm optimization variant with an inner variable learning strategy. *The Scientific World Journal, 2014*, 1–15.

Xie, W., Yu, W., & Zou, X. (2012). Diversity-maintained differential evolution embedded with gradient-based local search. *Soft Computing*, 1–25.

Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*, 82–102.

Zhan, Z. H., Zhang, J., Li, Y., & Chung, H. S. H. (2009). Adaptive particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, 39*, 1362–1381.

Zhan, Z. H., Zhang, J., Li, Y., & Shi, Y. H. (2011). Orthogonal learning particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 15*, 832–847.

Zhang, W. J., & Xie, X. F. (2003). DEPSO: Hybrid particle swarm with differential evolution operator. *In IEEE International Conference on Systems, Man and Cybernetics, 4*, 3816–3821.