Force-imitated Particle Swarm Optimization Using the Near-Neighbor Effect for Locating Multiple Optima

Lili Liu[*,a,b], Shengxiang Yang[c], Dingwei Wang[a,b]

[a] *College of Information Science and Engineering, Northeastern University Shenyang 110004, China*
[b] *Key Laboratory of Integrated Automation of Process Industry (Northeastern University), Ministry of Education, Shenyang 110004, China*
[c] *Department of Information Systems and Computing, Brunel University Uxbridge, Middlesex UB8 3PH, United Kingdom*

Abstract

Multimodal optimization problems pose a great challenge of locating multiple optima simultaneously in the search space to the particle swarm optimization (PSO) community. In this paper, the motion principle of particles in PSO is extended by using the near-neighbor effect in mechanical theory, which is a universal phenomenon in nature and society. In the proposed near-neighbor effect based force-imitated PSO (NN-FPSO) algorithm, each particle explores the promising regions where it resides under the composite forces produced by the "near-neighbor attractor" and "near-neighbor repeller", which are selected from the set of memorized personal best positions and the current swarm based on the principles of "superior-and-nearer" and "inferior-and-nearer", respectively. These two forces pull and push a particle to search for the nearby optimum. Hence, particles can simultaneously locate multiple optima quickly and precisely. Experiments are carried out to investigate the performance of NN-FPSO in comparison with a number of state-of-the-art PSO algorithms for locating multiple optima over a series of multimodal benchmark test functions. The experimental results indicate that the proposed NN-FPSO algorithm can efficiently locate multiple optima in multimodal fitness landscapes.

Key words: Particle swarm optimization, multimodal optimization

[*]Corresponding author.
Email address: `liulili1202@gmail.com` (Lili Liu)

problem, near-neighbor effect, force-imitated particle dynamics.

1. Introduction

Particle swarm optimization (PSO) is a branch of computational intelligence, inspired by social interaction among entities, rather than simply depending on separate individual cognitive abilities [17, 18]. Traditionally, studies on PSO algorithms have been concentrated on uni-modal optimization functions. For uni-modal optimization functions, the aim is to design algorithms that can quickly and precisely find a single global optimum in the solution space. PSO has been widely applied for solving various uni-modal optimization problems with promising results due to the property of fast convergence [31, 40].

However, most real world optimization problems are subject to multimodal environments, where multiple peaks exist in the fitness landscape [14, 19]. For multimodal optimization problems (MOPs), it is usually desirable to simultaneously locate multiple optima, including global optima or even local optima. This may be beneficial in terms of algorithm design and practical applications [29]. Regarding the first aspect, several meta-heuristic algorithms, including evolutionary algorithms (EAs) and swarm intelligence (SI), tend to converge toward the best solution found so far, while neglecting other promising areas that may have been previously explored. Striving to find multiple peaks allows the algorithm to search for different potential areas, and, hence, avoid getting stuck onto a local optimum during the optimization process. Regarding the second aspect, in various real world applications, the optimization tasks require to find more than one optima. Hence, searching for a diverse set of equally acceptable and high quality solutions may provide alternative solutions for an ideal decision-making in the macrocosmic sense [15]. In addition, many optimization problems in the real world are subject to dynamic environments, which also requires searching for multiple optima in order to rapidly detect environmental changes and respond to the changes to track the changing optima [5, 28].

Within the classical PSO model, the principles that govern each particle's movement are the interaction among particles and the retrospection of the past experience of the particle. Instead of running a classical optimization method (i.e., point-by-point approach) for many times, each of which aims to find one optimum, population-based techniques may be applied to find a set of optimal solutions in parallel in one run. Furthermore, due to the special feature of adjusting an individual's position according to the valuable information obtained through its neighbor connections, PSO turns out to be potentially attractive for solving MOPs. However, the classical PSO algorithm needs to be adapted for optimal results in finding multiple optima. The key drawback of the classical PSO algorithm lies in the diversity loss: once the swarm has converged to one point in the search space, particles may lose the ability to explore other optima [14, 36, 39].

In recent years, PSO has been applied to address MOPs with some promising results. Several approaches, such as the niching and speciation methods [1, 8, 20], have been developed for locating multiple optima in the fitness landscape. The main idea behind these approaches is to partition the whole swarm into several sub-swarms to locate different promising areas in the search space. With these approaches, two issues should be addressed. The first one is how to determine the neighbors for a particle, with the aim of assigning particles to suitable regions among different promising areas. The second issue concerns how to design the information-sharing strategy between a particle and its meaningful neighbors. The main task of the above two concerns usually requires a niching parameter to specify the radius of each niche. EAs have been proved sensitive to this user-specified parameter. However, setting it to a proper value is a major challenge since the knowledge may be unavailable a priori in real world applications [11, 30].

In this paper, a new force-imitated dynamics is introduced for the movement of particles in PSO. Inspired by a conventional phenomenon in nature, called the *near-neighbor effect*, each particle employs two force agents, the "near-neighbor attractor" and "near-neighbor repeller", which are selected from the memorized personal best sets and the current swarm according to the principles of "superior-and-nearer" and "inferior-and-nearer", respectively. These two members enforce the particle to move towards an appropriate area among different promising regions, and, hence, encourage the whole swarm population to locate various peaks in the multimodal fitness landscape. The proposed algorithm is called the near-neighbor effect based force-imitated PSO (NN-FPSO). NN-FPSO removes the need to specify the niching parameter. In order to investigate the performance of the proposed NN-FPSO algorithm for MOPs, experiments are carried out to analyze the effect of crucial techniques on the behavior of NN-FPSO and compare it with several state-of-the-art PSO algorithms on a set of commonly used benchmark MOPs in this paper.

The rest of this paper is outlined as follows. The next section presents relevant work, including the mathematical description of MOPs, the force based principles of the PSO algorithm, and a brief review on related researches for MOPs. Section 3 describes the inspiration from nature of this study, including the near-neighbor effect in nature and the force-imitated dynamics equation, which is derived from the force theory. Section 4 provides the proposed NN-FPSO in detail. The experimental study that provides an analysis over the effect of specific mechanisms and parameters, and the performance comparison of NN-FPSO with other PSO algorithms for locating multiple optima on MOPs, is presented in Section 5. Finally, Section 6 concludes this paper with some discussions on the relevant future work.

2. Related work

2.1. Multimodal optimization problems (MOPs)

Many commercial and engineering optimization problems have multiple optimal solutions, including global and local optima. The MOP is NP-hard in terms of its computational complexity [29]. Without loss of generality, we discuss maximization problems in this paper. Given a continuous domain D, which is a subset of the universe R_n, and a function $f : D \to R_n$, a global optimum $\vec{x}^*_G$ of f over the domain D is defined as any point $\vec{x}$ from D that satisfies the following condition:

$$\forall \vec{x} \in D, f(\vec{x}) \leq f(\vec{x}^*_G). \tag{1}$$

Given $\epsilon > 0$, and the ϵ-neighborhood of $\vec{x}^*$ can be defined as follows:

$$N(\vec{x}^*, \epsilon) = \{\vec{x} \mid \|\vec{x} - \vec{x}^*\| < \epsilon\} \tag{2}$$

Then, a local optimum $\vec{x}^*_L$ of f over the domain D is defined as any point $\vec{x}$ from D that satisfies the following condition:

$$\forall \vec{x} \in N(\vec{x}^*_L, \epsilon) \cap D, f(\vec{x}) \leq f(\vec{x}^*_L). \tag{3}$$

Optimization algorithms usually suffer from some difficulties in solving MOPs due to the existence of multiple global optima. For example, Figure 1 shows the fitness landscape of a typical multimodal function, the 2-dimensional Shubert function, denoted *Shubert 2D* in this paper. Shubert 2D has 760 optima, including 18 global optima. As shown in Figure 1, there

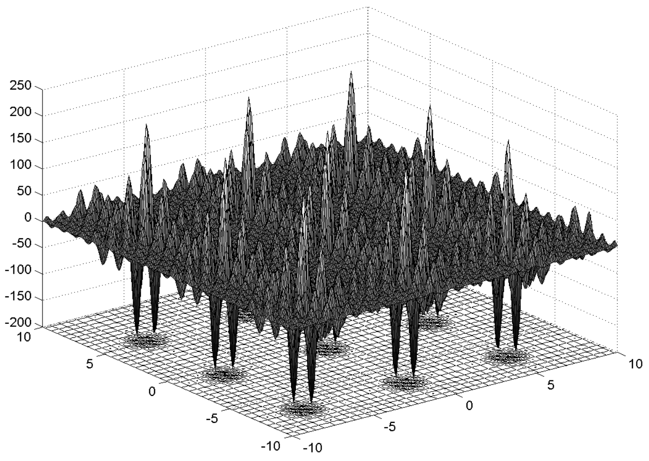


Figure 1: The 2-dimensional Shubert function.

exist many potential attractors distributed in different regions of the fitness landscape. It is a difficult task to distinguish among them during the whole optimization process. Therefore, it may be beneficial to evolve sub-groups in parallel, which are driven by suitable attractors, separately.

In practical applications, it is usually desirable to detect all the global optima, such as computing all *Nash equilibria* for producing a reliable estimation of the outcome that can be reached through a game playing. Another interesting research area is the computation of period orbits of nonlinear mappings [29], and detecting all points of some specific types can be used in dissipative dynamical systems.

2.2. Particle swarm optimization (PSO)

PSO was first introduced by Kennedy and Eberhart in [13, 16]. PSO employs a population of particles that fly over the fitness landscape, of which the swarm dynamics was inspired by the collective behavior of organisms, such as bird flocking and fish schooling. Each particle holds a memory of the best position that it has seen so far and the best position obtained within its neighborhood. A particle updates its velocity based on its current velocity and position along with the above two memorized positions. The modification of the moving orbit of a particle, say particle i, is described as follows [9]:

$$\vec{v}_i(t+1) = \chi\vec{v}_i(t) + c_1\vec{\xi} \times (\vec{p}_i(t) - \vec{x}_i(t)) + c_2\vec{\eta} \times (\vec{p}_g(t) - \vec{x}_i(t)) \qquad (4)$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1), \tag{5}$$

where χ is the inertia weight, which controls the degree that the velocity of a particle at time t influences the velocity of that particle at time $t+1$. Vectors $\vec{v}_i(t)$ and $\vec{x}_i(t)$ represent the current velocity and position of particle i at time t, respectively, $\vec{p}_i(t)$ and $\vec{p}_g(t)$ represent the position of the best solution discovered so far by particle i and the position discovered so far by all particles in the neighborhood of particle i, respectively, c_1 and c_2 are the acceleration constants that determine the influence of the two attractors to particle i, respectively, and $\vec{\xi}$ and $\vec{\eta}$ are random vectors with each constituent randomly drawn with a uniform distribution from [0, 1].

In the classical PSO model, according to Eqs. (1) and (2), particles share information through the swarm attractor, $\vec{p}_g$, and evoke memories via particle attractors, $\vec{p}_i$. Based on Eqs. (1) and (2), each particle's movement can also be considered as being accelerated by two elastic forces produced by two attractors, which correspond to the previous best position found by itself and the previous best position found by particles in its neighborhood, respectively. The velocity update for a particle, say particle i, can be modified as follows:

$$\vec{v}_i(t+1) = \chi(\vec{v}_i(t) + \vec{a}_i(t)) \tag{6}$$

where χ and $\vec{v}_i(t)$ are as defined before, and $\vec{a}_i(t)$ is the acceleration produced by the force described as follows:

$$\vec{a}_i(t) = \sum_{j \in A_i(t)} \frac{F_{ij}(t)}{m_i(t)} \tag{7}$$

where $A_i(t)$ is the set of agents that act on agent i at time t, $F_{ij}(t)$ is the force produced by agent j on agent i at time t, and $m_i(t)$ is the mass of agent i at time t. In this study, we assume $m_i(t)$ is a constant with the value 1. For the classical PSO model, we have the following acceleration formula:

$$\vec{a}_i(t) = c_1\vec{\xi} \times (\vec{p}_i(t) - \vec{x}_i(t)) + c_2\vec{\eta} \times (\vec{p}_g(t) - \vec{x}_i(t)) \tag{8}$$

where $\vec{p}_i(t)$, $\vec{p}_g(t)$, c_1, c_2, $\vec{\xi}$, and $\vec{\eta}$ are as defined before.

Several versions of the PSO algorithm have been developed [31] since PSO was first introduced. Among them, there are two main models, called *gbest* (global best) and *lbest* (local best), respectively. These two models differ in the way of defining the neighborhood for each particle. In the *gbest* model, the neighborhood of a particle consists of the particles in the whole swarm, which share information between each other. On the contrary, in the *lbest* model, the neighborhood of a particle consists of several fixed particles.

Over the years, researchers have also investigated the force-based dynamics for the swarm in PSO, such as controlling the movement of particles, and evolving with genetic programming through a series of force generating equations [29]. Blackwell [4] used a charged PSO algorithm to solve the dynamic optimization problem by considering a bi-modal parabolic environment of high spatial and temporal severity. This work was extended by using quantum swarms and charged particles to avoid the collision of particles for addressing dynamic environments [5]. The idea of applying attraction and repulsion in PSO was proposed by some researchers. Riget and Vesterstroem have proposed an attractive and repulsive PSO (ARPSO) algorithm with the aim of preventing premature convergence on MOPs. Within ARPSO, the attraction and repulsion phases are two stages when updating the velocity of a particle. The swarm is alternated between these two phases according to a diversity measurement. The attraction phase causes the particles converging toward one another, while repulsion pushes particles away from one another for enhancing diversification. Dalland and Lam [10] proposed a modified ARPSO algorithm and implemented it on a combinatorial benchmark problem, the orienteering problem, which is a variation of the well-known traveling salesmen problem.

2.3. Researches on locating multiple optima

In recent years, PSO has been increasingly applied for MOPs. When solving MOPs, an efficient PSO algorithm should be able to locate and maintain multiple peaks simultaneously [21, 29]. This greatly challenges classic PSO algorithms due to the diversity loss.

Regarding locating multiple optima in multimodal landscapes, the behaviour of the two PSO models, i.e., the *gbest* model and the *lbest* model, is quite different. For the *gbest* PSO model, van den Bergh and Engelbrecht [40] proved that particles will congregate onto a single point, even incapable of obtaining different global optima when the algorithm is executed for many times. Parsopoulos and Vrahitis [29] noted that when applying the *gbest* model for tackling MOPs, the swarm tends to move back and forth in the landscape, which causes particles fail to decide where to land. With the idea of isolating a potentially good solution and stretching the fitness landscape for a comprehensive exploration for other potential global optima, the algorithm has been proved to be able to locate all optima on the test functions. The technique is similar to the derating method applied in the sequential niche method from the GA community [1].

The *lbest* PSO model is less vulnerable to the attraction of a global or local optimum than the *gbest* model. For the *lbest* PSO model, Parsopoulos [29] has empirically indicated that it is inefficient for locating optima because the adopted neighbor best is not suitable for attraction. Brits *et al.* [8] have proposed a *nbest* PSO algorithm using the "neighborhood best" for each particle, which is defined as the center of positions of its n closest members. However, the method suffers from the difficulty of setting the parameter n, which requires a prior knowledge about a specific problem domain.

The niching method, which was inspired by the social interaction and adaption of individuals around multiple resources, is an efficient technique for EAs to address MOPs. The basic principle of the niching method is to maintain the population diversity by dividing the population into several sub-groups in order to focus on different promising areas in the search space, thereby finding multiple peaks in parallel. Various niching techniques have been proposed in the EA community. A brief review is provided as follows.

Petrowski [30] introduced a niching method, called *clearing*, for genetic algorithms (GAs). The motivation of this method is to share finite resources in the sub-population that contains similar individuals, which are measured by the Euclidean distance. The fitness of inferior members within a pre-specified radius around a dominant member is set to zero, which is considered to be "cleared". This technique has shown a competitive ability of avoiding gene drift.

Similar to the above method, the speciation approach is also a considerable technique for locating multiple optima. Li [19] has explored the clearing method by using the notion of speciation to GAs. Different from Petrowsk's method, the speciation method discourages the interaction of particles in different sub-groups, and tries to simultaneously locate multiple optima in the search space. Through the adaptive creation of multiple species, Li et al. [20, 21] have further incorporated the idea of species into the classical PSO algorithm and differential evolution (DE) for solving MOPs. In their algorithm, at each iteration, multiple species are identified, and a neighbor best particle, called *species seed*, is determined within each specie. Their algorithms have shown to be more competitive on MOPs than those reported in the literature.

The crowding method was originally introduced by De Jong [12] as a diversification technique. It was inspired by the phenomenon that similar members compete for limited resources. In the crowding method, different promising areas are occupied by dissimilar individuals, while similar population members compete within the same niche. Within a niche, inferior old members will be replaced (i.e., crowded out of the population) by the fitter ones when the niche has reached its maximal capacity. Mahfound [23, 24] developed some novel methods to eliminate the requirement of user-specified parameters and reduce the replacement error for matching multiple peaks. With the aim of providing a restorative pressure for GAs, the probabilistic crowding proposed by Mengshoel [27] has shown some promising results in multimodal fitness landscapes.

Goldberg and Richardson [14] suggested a fitness sharing method with the purposes of locating and preserving multiple optima. This method allows the growth of similar individuals by enabling each population member to share its fitness assignment with nearby members, where the proximity between two individuals is synthetically considered with their fitness and distance. Deb and Goldberg [11] further studied effective techniques to set the niche parameters to improve the behavior of locating multiple optima.

As a PSO based niching technique, NichePSO [8] identifies niches by measuring changes of the particle's fitness. A sub-swarm is created when a particle's fitness exhibits a little variance over several iterations, which contains this particle and its closest topological neighbors. NichePSO has shown to be substantially successful in finding multiple optima, including global and some local optima, on MOPs.

Researches have also been carried out on enhancing the robustness of niching techniques for MOPs. It is a crucial task to determine the niching radius to estimate the distance between those multiple optima without prior information of a problem. Bird [2] presented adaptive niching PSO algorithms that adaptively adjust the radius without per-specifying this parameter. The experimental results indicate that this algorithm is insensitive to the adopted parameters, making it capable of solving a series of problems without additional tunings.

A recent work by Li [22] aimed to remove the requirement for specifying niching parameters. The idea is to encourage each particle to absorb information from its fitter-and-closer neighbor, which provides PSO with a good performance on locating global optima. For each particle, the neighbor that holds the largest ratio of the fitness difference and the Euclidean distance is selected from the memory swarm. Their experimental results indicate that sub-groups of particles near each other are able to explore multiple local peaks in MOPs and that this algorithm is suitable for finding optima in the multimodal landscape in parallel when the population size is sufficiently large.

The aforementioned approaches indicate some key considerations to improve the performance of PSO in multimodal fitness landscapes. They are summarized as follows:

1. Particles should exploit information from their respective neighbors that hold valuable knowledge. This may pull the particles to search for promising regions. On the other hand, each particle should take advantage of knowledge from its weaker neighbors. This may push the particle to move along in the probing direction and extensively explore the search space.
2. Dynamic communication models based on the feedback of the fitness landscape may be beneficial for the adaptive searching and locating behavior.
3. Personal best so far positions of particles can provide reliable and valuable information found so far by the population, which can create an interaction network to correctly guide the movement of particles.
4. It would be desirable to remove the need of pre-specified parameters without any prior information of a problem.

In the following section, we describe the inspiration from nature regarding two principles, the near-neighbor effect and the force-imitated mechanism. Then, the proposed NN-FPSO algorithm that applies the force-imitated mechanism using the near-neighbor effect will be described in detail in Section 4.

3. Inspiration from Nature

3.1. The near-neighbor effect

The concept of the near-neighbor effect (i.e., the vicinal effect) is derived from nature. It usually refers to an interactive phenomenon between two or multiple individuals, which hold similar characters, structures, or adjacent locations, such as the attraction and repulsion effect on the vicinal group in the chemology [32], and the next-nearest-neighbor hopping of the crystal lattice in the physical domain [38]. A common animal behavior is that an individual tends to perceive and react to what it sees locally, and is most likely to be attracted by the superior ones while attempting to depart from the inferior ones [26, 41]. These mechanisms enable each organism to process some promising properties through the agency of its immediate surroundings (i.e., neighbors), and hence magnifies its influence across the whole population or flock.

There are two essential principles behind the near-neighbor effect phenomenon in physics. They are briefly summarized as follows:

1. Selectivity: The activation of the near-neighbor effect is determined by the surface or properties of an organism. In general, this interaction often takes place between neighbors in the spatial measurement with consistent attributes or characters in the intrinsic aspects.
2. Pluralism: There are numerous categories of these action forms. The pattern (either mono-directional or bi-directional) and degree of action are also determined by the properties of organisms.

3.2. The force-imitated mechanism

In the physics domain, force is the primary reason that causes system dynamics, which is carried out by the quanta of interactions. The motion state of individuals will be changed when they are subject to external forces. The "action in distance" force, such as the universal gravitation and the Coulomb force, acts between two separated entities within a certain range without any delay. The force is proportional to a certain operation between properties of these two entities (i.e., the mass and the electrical quantity), while is inversely proportional to the square of the distance between them [33]. Therefore, at a specific time t, the force acting on agent i at position $\vec{x}_i(t)$ from agent j at position $\vec{x}_j(t)$ can be described as follows:

$$\vec{F}_{ij}(t) = K(t)\frac{c_i(t) \bigotimes c_j(t)}{D_{ij}^n(t)}(\vec{x}_i(t) - \vec{x}_j(t)) \tag{9}$$

where $K(t)$ is a constant at time t, $D_{ij}^n(t) = \|\vec{x}_i(t) - \vec{x}_j(t)\|^n (n = 1, 2, 3, \ldots)$, $c_i(t)$ and $c_j(t)$ are the concerned attribute (such as the mass in the Newton's law of universal gravitation [32] and the electrical charge in the Coulomb's law [38]) of agent i and agent j, respectively, the symbol "$\bigotimes$" represents a certain mathematical operation between the two adopted attributes. In this study, Eq. (9) is called the *force-imitated equation.*

4. The proposed NN-FPSO algorithm

In this paper, the concept of the near-neighbor effect is integrated into the force-imitated mechanism and then applied into the PSO algorithm to improve its ability of locating multiple optima on MOPs, which results in the proposed NN-FPSO algorithm in this paper. NN-FPSO adopts a new information-sharing scheme, which is described as follows. Each particle iteratively samples a region conducted by a compound acceleration vector, which is composed of three constituents: the difference between the particle and its personal best position, an attraction acceleration, and a repulsion acceleration. The attraction acceleration is produced by the near-neighbor attractor selected from the memory swarm based on a spatial method, which allows the particle to explore the possible region where a near optimum may reside. At the meanwhile, in order to avoid a particle from being trapped into the current promising area (e.g., being stuck onto a local best point) and, hence, losing the ability to search for other peaks, a near-neighbor repeller is selected from the current swarm, also based a spatial technique. By applying this method at each iteration, the near-neighbor attractor and the near-neighbor repeller for each particle are identified and drive that particle to search for different optima in its neighborhood.

Following the above discussions, the acceleration formula in NN-FPSO is changed from Eq. (8) in the classic PSO model to the following:

$$\vec{a}_i(t) = c_1\vec{\xi} \times (\vec{p}_i(t) - \vec{x}_i(t)) + \vec{a}_{i\_att}(t) + \vec{a}_{i\_rep}(t) \tag{10}$$

$$\vec{a}_{i\_att}(t) = \vec{F}_{i\_att}(t)/m_i(t) \tag{11}$$

$$\vec{a}_{i\_rep}(t) = \vec{F}_{i\_rep}(t)/m_i(t) \tag{12}$$

where $\vec{a}_{i\_att}(t)$ and $\vec{a}_{i\_rep}(t)$ are the acceleration produced by the near-neighbor attraction force $\vec{F}_{i\_att}(t)$ and the near-neighbor repulsion force $\vec{F}_{i\_rep}(t)$ on particle i at iteration t, respectively. Figure 2 illustrates the possible movement trend of a particle under these two forces, and presents the comparison between the classical PSO algorithm and the proposed NN-FPSO algorithm in terms of where a particle may move to a possible region.

It is noticeable that the particle $\vec{x}_i$ tends to explore in a new neighborhood region under the guidance of valuable information, including the "superior" information derived from the "near-neighbor" best solution in the memory

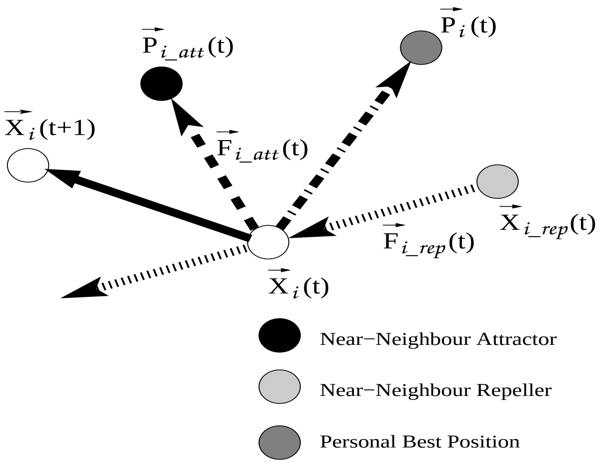


Figure 2: The update of particle i from $\vec{x}_i(t)$ to $\vec{x}_i(t+1)$ driven by the composite force produced by the near-neighbor attractor $\vec{F}_{i\_att}(t)$, the near-neighbor repeller $\vec{F}_{i\_rep}(t)$, and the personal best $\vec{P}_i(t)$

swarm, denoted as $\vec{P}_{i\_att}(t)$, and the "inferior" information derived from the worst solution in the current swarm, denoted as $\vec{X}_{i\_rep}(t)$. This way, this technique may help the population search comprehensively in the search space while guaranteeing the searching speed.

4.1. The near-neighbor attraction force

In order to encourage each particle to search for the "achievable" optimum of its own neighborhood in the multimodel fitness landscape, the near-neighbor attraction force is introduced. The force-loaded particle is adopted according to a spatial method. The near-neighbor attraction force is calculated based on Eq. (9) described in Section 3 as follows:

$$\vec{F}_{i\_att} = K_{att}\frac{f(\vec{P}_{i\_att}) - f(\vec{P}_i)}{\|\vec{P}_{i\_att} - \vec{P}_i\|^2}(\vec{P}_{i\_att} - \vec{P}_i) \tag{13}$$

$$K_{att} = \frac{I_{att} \times \|A\|^2}{f(\vec{X}_{Best}) - f_(\vec{X}_{Worst})} \tag{14}$$

where $\vec{P}_i$ is the personal best position of particle i, $\vec{P}_{i\_att}$ is the near-neighbor attractor, I_{att} is the attraction influence factor, which implies the influence degree of the near-neighbor attractor of a particle on the movement trend of that particle, $\vec{X}_{Best}$ and $\vec{X}_{Worst}$ are the best solution and the worst solution of the current swarm respectively, and $\|A\|$ is the spatial size of the solution space and can be described as follows:

$$\|A\| = \sqrt{\sum_{k=1}^{d} (X_k^U - X_k^L)^2} \tag{15}$$

where d is the dimension of the search space and X_k^U and X_k^L are the upper and lower bound of the search space regarding the k-th dimension, respectively. The near-neighbor attractor $\vec{P}_{i\_att}$ can be determined as follows:

$$\vec{P}_{i\_att} = \underset{\vec{P}_j : j=1,\cdots,M \ \wedge \ j \neq i}{\operatorname{argmax}} \frac{f(\vec{P}_j) - f_(\vec{P}_i)}{\|\vec{P}_j - \vec{P}_i\|} \tag{16}$$

where M is the swarm size.

We adopt such a rule for the near-neighbor attraction force for two reasons. First, the memory swarm is much more stable than the current swarm. This is because the best points found so far by corresponding moving particles are updated only when new better points are found [22]. Second, each particle can absorb the "receivable" and "superior" information. On the one hand, the similar information encourages a particle to efficiently utilize the information of its own and its meaningful neighbors, and, hence, move toward the peak in the neighborhood. On the other hand, the superior information provides particles with more possible regions where different optima may reside.

4.2. The near-neighbor repulsion force

One potential issue regarding the efficiency of the NN-FPSO algorithm is that particles tend to converge to the local optima. Therefore, the sole attraction force does not contribute further to the improvement of exploration. In order to avoid particles from being trapped onto the local optima, the near-neighbor repulsion force is introduced into NN-FPSO and can be calculated as follows:

$$\vec{F}_{i\_rep} = K_{rep} \frac{f(\vec{X}_{i\_rep}) - f(\vec{x}_i)}{\|\vec{X}_{i\_rep} - \vec{x}_i\|^2} (\vec{X}_{i\_rep} - \vec{x}_i) \tag{17}$$

$$K_{rep} = \frac{I_{rep} \times \|A\|^2}{f(\vec{X}_{Best}) - f_(\vec{X}_{Worst})} \tag{18}$$

where I_{rep} is the repulsion-influence factor, which implies the influence degree of the near-neighbor repeller of a particle on the movement trend of that particle, $\|A\|$ is the spatial size of the solution space, and $\vec{X}_{i\_rep}$ is the near-neighbor repeller of particle i, which can be determined as follows:

$$\vec{X}_{i\_rep} = \underset{\vec{X}_j : j=1,2,\cdots,M \ \wedge \ j \neq i}{\text{argmax}} \frac{f(\vec{x}_i) - f(\vec{X}_j)}{\|\vec{x}_i - \vec{X}_j\|} \tag{19}$$

The reason we adopt such a principle lies in that it is expected to make a better use of the inferior-and-nearer particles by exploiting some indicative information so that other areas of the search space can be fully explored.

It is noticeable that the idea of applying attraction and repulsion in PSO has been introduced in ARPSO, which has shown competitive performance in multimodal contexts. As mentioned above, the basic motivation of ARPSO is to use a diversity measure to alternate the algorithm between exploration and exploitation behaviors, and hence, encouraging individuals to cover more regions for locating the global optima. Different from ARPSO, NN-FPSO uses the interaction between particles and their meaningful neighbors, which is expected to drive individuals to their suitable regions among different promising regions where the global (local) optima may reside [19]. The near-neighbor effect is applied for developing multiple partitions of the whole population in parallel for locating optima in the multimodal fitness landscape.

5. Experimental study

5.1. Test functions

In order to test the performance of investigated algorithms, seven benchmark multimodal functions were used in the experimental study. They represent MOPs of different properties. More detailed description of these functions can be found in [21, 25, 35]. The first five functions $F1$ to $F5$ are shown in Table 1. $F1$ to $F4$ are relatively simple MOPs, which are low-dimensional functions. $F5$ is the Shubert function, which is a much more challenging function as it is highly multimodal. For example, $F5$ in the 2-dimensional space, denoted Shubert 2D or $F5(2D)$, is as shown in Figure 1, where the 18 global optima out of 760 optima form 9 pairs of two global peaks that

Table 1: Multimodal test functions

Function	Range	Number of Global Peaks
$F1(x,y) = (y - \frac{5.1x^2}{4\pi^2} + \frac{5x}{\pi} - 6)^2$	$-5 \leq x \leq 10$	5
$+10(1 - \frac{1}{8\pi})cos(x) + 10$	$0 \leq y \leq 15$	
$F2(x,y) = 200 - (x^2 + y - 11)^2$	$-6 \leq x, y \leq 6$	4
$-(x + y^2 - 7)^2$		
$F3(x,y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2$	$-1.9 \leq x \leq 1.9$	2
$+xy + (-4 + 4y^2)y^2]$	$-1.1 \leq y \leq 1.1$	
$F4(x) = sin^6(5\pi x)$	$0 \leq x \leq 1$	5
$F5(\vec{x}) = \prod_{i=1}^{d} \sum_{j=1}^{5} j \times cos[(i+1)x_i + j]$	$-10 \leq x_i \leq 10$	$d \cdot 3^d$
$F6(\vec{x}) = \frac{1}{d} \sum_{i=1}^{d} sin(10log(x_i))$	$0.25 \leq x_i \leq 10$	6^d

are the closest to each other. $F5$ poses a great challenge to the niching technique since the niching radius is unique for all catchment areas. The number of global optima in $F5$ increases quickly with the increment of the number of dimensions, i.e., the value of d. For example, the 3-dimensional Shubert function, denoted Shubert 3D or $F5(3D)$, has 81 global optima in 27 groups and the 4-dimensional Shubert function, denoted Shubert 4D or $F5(4D)$, has 324 global optima in 81 groups.

The sixth benchmark problem $F6$ is the invented Vincent function, which has 6^d global peaks. Different from the regular distances between global peaks in $F5$, it has no local peaks but has vastly different spacing between global optima. The seventh test function $F7$ is the Hump function with an arbitrary number of peaks, which has been widely used as a multimodal benchmark problem [35]. Within this problem, each optimum can be represented by four features: the location, the height, the shape, and the radius of the basin of attractors. The hump function with K peaks is defined as follows:

$$f(\vec{x}) = \max_{i=1,2,\cdots,K} f_i(\vec{x}) \tag{20}$$

$$f_i(\vec{x}) = \begin{cases} H_i[1 - (\frac{d(\vec{x},\vec{X}_i)}{R_i})^{\alpha_i}], & \text{if } d(\vec{x},\vec{X}_i) \leq R_i \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

where $f_i(\vec{x})$ is the definition of peak i, $\vec{X}_i$, H_i, and R_i are the location, height, and radius of peak i, respectively, α_i is the shape factor of peak i, and $d(\vec{x}, \vec{X}_i)$ is the Euclidean distance between solution $\vec{x}$ and the center of peak i.

5.2. Experimental design

In this section, three sets of experiments are carried out to investigate the performance of NN-FPSO in multimodal environments. In the first set of experiments, the sensitivity analysis on the effect of key parameters and critical approaches on the performance of NN-FPSO is carried out on different multimodal test functions. In addition, a guideline for setting the parameters introduced into NN-FPSO (i.e., I_{att} and I_{rep}) is provided, with an experimental verification that NN-FPSO is robust regarding them within a range for different MOPs. In the second set of experiments, NN-FPSO is compared with ARPSO, which has also used the idea of attraction and repulsion in PSO on the test functions. In the third set of experiments, NN-FPSO is compared with four other peer PSO algorithms in multimodal environments with different complexities of the fitness landscape.

The four peer PSO algorithms are described as follows. The first PSO algorithm is the species-based PSO (SPSO) algorithm proposed in [20]. SPSO uses a local "species seed", which provides the local best to particles whose positions are within a specific species radius r_s. SPSO has been shown to produce competitive results in multimodal environments and is more effective than other niching techniques in the literature when the species radius is set to a proper value. The second peer PSO algorithm is the fitness Euclidean ratio PSO (FER-PSO) algorithm [22]. FER-PSO follows the basic idea of encouraging the survival of fitter and closer particles and removes the need to specify any niching parameters for locating multiple optima. The third peer algorithm is the adaptive niching PSO (ANPSO) algorithm proposed in [2] with the aim of releasing the requirement for pre-determining any niching parameters. ANPSO is another state-of-the-art PSO model for locating multiple optima in parallel. The fourth peer algorithm is the enhanced SPSO (ESPSO) algorithm [3], whose primary motivation is also to reduce the sensitivity of SPSO to the niching parameters.

In the experiments, for each PSO variant, the learning factors $c_1 = c_2 = 2.05$ and the inertia weight $\omega = 0.729844$ were applied as suggested in [20]. For the peer PSO algorithms, other parameters were set to the values that were recommended by their authors respectively. For each experiment of an algorithm on a test problem, 50 independent runs were executed with the same set of random seeds in order to have a fair comparison among different algorithms.

In order to evaluate the ability of algorithms for locating all global optima in parallel, three performance measurements that were used in [3] were also adopted in this study. They are the *accuracy*, the *convergence speed*, and the *success rate*, which are described as follows.

5.2.1. Performance measurements

The first performance measurement is the *accuracy*, denoted as μ in this paper. It is defined as the average of fitness differences between all known global optima and their closest particles. Accuracy was recorded in the final iteration step and calculated as follows [2]:

$$\mu = \frac{1}{N_{opt}} \sum_{j=1}^{N_{opt}} |f(\vec{opt}_j) - f(\vec{x}_{opt_j})| \tag{22}$$

where N_{opt} is the number of all known global optima, $\vec{opt}_j$ is the j-th global optimum, and $\vec{x}_{opt_j}$ is the corresponding particle that is the closest to $\vec{opt}_j$. It can be seen that this measurement can give an accurate indication of how closely an algorithm identifies all global optima.

In order to test the ability of algorithms to quickly and precisely converge onto all global optima, the *convergence speed* at a required level of accuracy is taken as the second performance measurement in this paper. The fitness difference between each known global optimum $\vec{opt}_j$ and its closest particle $x_{\vec{opt}_j}$ is calculated for each iteration. A global optimum is considered to be found if a solution is close enough to the global optimum according to an expected accuracy acceptance threshold δ $(0 \leq \delta \leq 1)$, that is, the following condition should be met:

$$\forall \vec{x} \in S_{opt}\ \exists \vec{y} \in S : min\{\|\vec{x} - \vec{y}\|\} \wedge \|f(\vec{x}) - f(\vec{y})\| \leq \delta \tag{23}$$

where S_{opt} is the set of all known global optima, S is the current swarm, and $min\{\|\vec{x} - \vec{y}\|\}$ returns the closest pair of a global optimum and a solution.

The above equation can be used to calculate the number of iterations C_{Step} required by an algorithm to find all global optima, i.e., the convergence speed of an algorithm, and the number of evaluations (C_e) for an algorithm to locate all global optima can be calculated as follows:

$$C_e = C_{Step} \times M \tag{24}$$

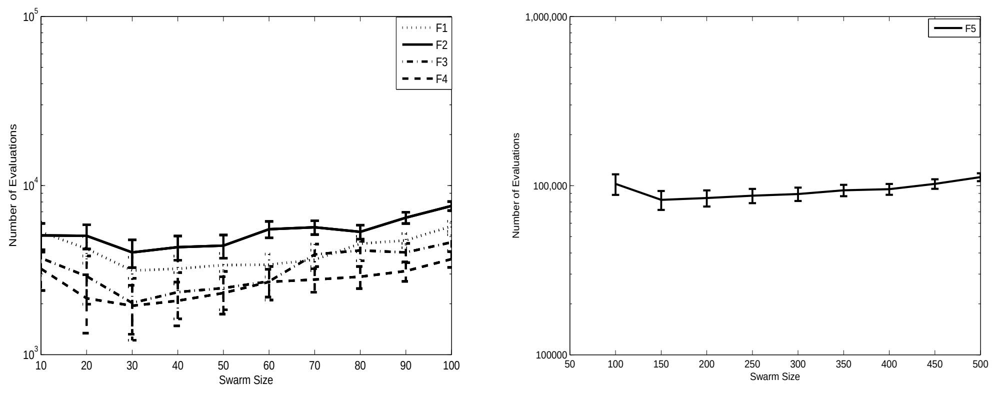
where M is the swarm size. Eq. (23) can also be used to justify the number of peaks found by an algorithm within a maximum allowable number of iterations.

In addition to the above two performance measurements, the performance measurement of the *success rate* was also used in the experimental study, which represents the percentage of runs in which all global optima were successfully located.

5.3. Sensitivity analysis of parameters of NN-FPSO

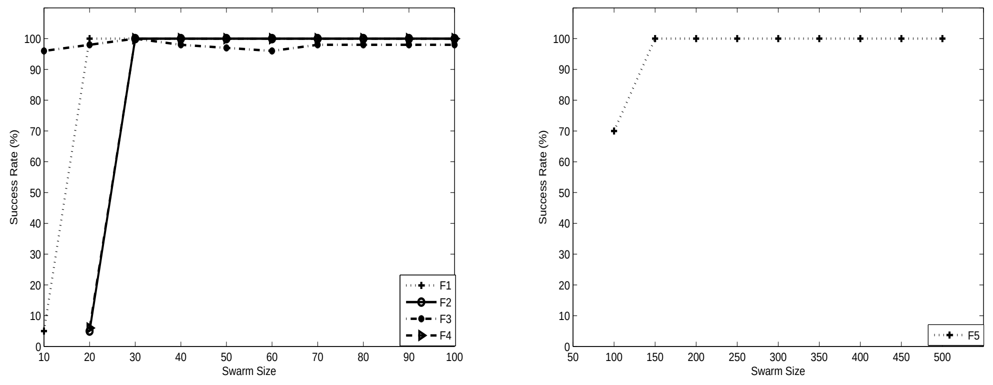
5.3.1. Effect of the swarm size

The aim of this set of experiments is to test the effect of the swarm size M on the performance of NN-FPSO on different MOPs. Given that the complexity of the test functions is quite different, experiments were carried out with M set in the range of $[10, 100]$ and $\delta = 0.00001$ for $F1$-$F4$ and with M set in the range of $[50, 500]$ and $\delta = 0.1$ for $F5(2D)$ due to a relatively larger number of global optima in $F5$. Other parameters of NN-FPSO were set as follows: $I_{att} = 0.5$, $I_{rep} = 0.1$, and a maximum of 200000 evaluations was allowed for each run of NN-FPSO on a MOP. The experimental results regarding the convergence speed and the success rate averaged over 50 runs are provided in Figure 3 and Figure 4, respectively.



(a) (b)

Figure 3: The average number of evaluations needed to find all global optima over 50 runs of NN-FPSO with $I_{att} = 0.5$, $I_{rep} = 0.1$, and different swarm sizes on functions: (a) F_1-F_4 and (b) $F_5(2D)$.



(a) (b)

Figure 4: The average success rate over 50 runs of NN-FPSO with $I_{att} = 0.5$, $I_{rep} = 0.1$, and different swarm sizes on functions: (a) F_1-F_4 and (b) $F_5(2D)$.

From Figure 3, it can be seen that for $F1$-$F4$ the swarm size around 30 always gives the best performance, and for $F5(2D)$ the best performance is achieved with the swarm size set to 150. The convergence speed decreases rapidly when the swarm size is below these two values in the above two cases, and a larger swarm size uses more evaluations per iteration and hence hinders the optimization performance. The standard deviation, as shown in Figure 3, decreases with an increasing swarm size in all investigated functions.

When examining the reliability of NN-FPSO for simultaneously locating all global optima, Figure 4 indicates that the swarm size does not have a large effect on the success rate, as long as it is sufficient for the population to cover peaks. NN-FPSO achieves a 100% success rate when the swarm size is above 30 for $F1$-$F4$ and above 100 for $F5(2D)$, respectively.

5.3.2. Effect of the near-neighbor attraction factor (I_{att})

As mentioned in Section 4, the near-neighbor attraction has a potential to drive a particle towards its fitter-and-closer neighboring point, and hence enables the particle to explore the possible area of its own surrounding. The attraction influence factor I_{att} has a significant influence on NN-FPSO's ability of locating multiple optima in multimodal environments. Setting I_{att} to a large value may cause particles ignore the accumulated information of its personal best (see Eq. (10)), and induce a risk of wasting the computational effort of a particle to search in the same area where its local fitter neighbor

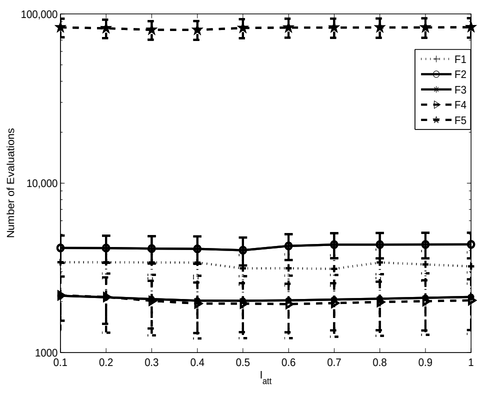
Table 2: The number of peaks found (mean and standard deviation over 50 runs) by NN-FPSO with $I_{rep} = 0.1$, $M = 30$, and different I_{att} on MOPs

I_{att}	F1	F2	F3	F4	F5(2D)	F5(3D)	F5(4D)
0	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00
0.1	**5.00±0.00**	**4.00±0.00**	1.98±0.45	4.85±0.25	**18.00±0.00**	**41.15±5.85**	**61.42±6.73**
0.5	**5.00±0.00**	**4.00±0.00**	**2.00±0.00**	**5.00±0.00**	17.94±3.12	40.98±4.28	59.89±4.92
1	4.87±0.48	3.82±0.47	1.92±0.21	4.75±0.13	16.93±2.81	38.24±3.12	52.48±3.12
5	1.25±0.33	2.37±0.24	1.24±0.35	3.28±0.25	8.48±2.52	12.27±2.15	14.15±4.24
10	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

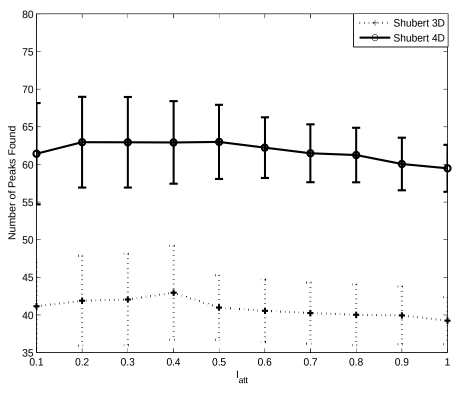
has already searched. On the contrast, setting I_{att} to a small value may weaken the capacity of particles to identify their own regions properly.

In order to test the influence of I_{att} on the performance of NN-FPSO, experiments were carried out with I_{att} set to different values in the set $\{0, 0.1, 0.5, 1, 5, 10\}$. In addition to $F1$-$F5(2D)$, the Shubert 3D and Shubert 4D functions were also used as the test functions. The swarm size of NN-FPSO was set to 30 for $F1$-$F4$ and 100 for $F5(2D)$. The swarm size was set to 1500 for Shubert 3D and Shubert 4D since they have a large number of optima in the search space. The parameter $I_{rep} = 0.1$ was used in NN-FPSO, and a maximum of 200000 evaluations was allowed for each run of NN-FPSO on a MOP. The experimental results regarding the number of peaks found by NN-FPSO within 200000 evaluations over 50 runs are shown in Table 2. In Table 2 (and other tables in this paper), the best result achieved for each function is shown in bold font.

The results in Table 2 show that the attraction influence factor I_{att} should be set in the range of $[0.1, 1.0]$ for a better performance of NN-FPSO on the test functions. It can be observed that, for a function with a relatively small number of peaks (i.e., $F1$-$F4$), setting I_{att} to around 0.5 always gives a better result than other settings. When the environment becomes more multimodal and challenging (i.e., Shubert 2D, Shubert 3D, and Shubert 4D functions), setting I_{att} to around 0.1 seems more suitable. This indicates that diversification is beneficial for handling those problems with relative more peaks in the multimodal fitness landscape. The poor performance for $I_{att} = 0$ indicates that the near-neighbor attraction scheme is beneficial for the performance of NN-FPSO on MOPs. When $I_{att} = 10$, which gives the largest attraction effect among all the settings of I_{att}, NN-FPSO always performs worse than other settings (except for the case of $I_{att} = 0$ as discussed above). This occurs



(a)



(b)

Figure 5: (a) The average number of evaluations needed to find all global optima on functions F_1-$F_5(2D)$ and (b) the average number of peaks found on Shubert 3D and Shubert 4D over 50 runs of NN-FPSO with $I_{rep} = 0.1$ and different I_{att}.

due to two factors. Firstly, too much attraction from the near-neighbor attractor weakens too much the effect of the personal experience of a particle, which may produce stable and valuable information for a better adaptation for multimodal landscapes. Secondly, the effect of the exploration of nearby possible regions discovered is also vital for the identification process. This indicates that a good trade-off between the search ability for identifying promising areas and the exploration ability for promoting successive movements is important for locating all global optima in parallel.

Since one key aim of this study is to minimize the requirement of tuning parameters and the problem dependency of the algorithm, it is necessary to test the robustness of the NN-FPSO algorithm regarding relevant parameters. Based on the above analysis, the experimental results regarding NN-FPSO's ability of locating global optima with $I_{rep} = 0.1$ and I_{att} set in the range of $[0.1, 1.0]$ are provided in Figure 5. From Figure 5(a), it can be seen that I_{att} in the range of $[0.1, 1.0]$ does not have a large effect on the number of evaluations required by NN-FPSO to locate all the global optima for $F1$-$F5(2D)$. Again setting I_{att} to around 0.5 can produce the best performance for $F1$-$F5(2D)$ and setting I_{att} to values smaller than 0.5 causes an increase in the number of evaluations required. The reason may lie in the fact that NN-FPSO with $I_{att} = 0.5$ has a sufficient search power to quickly locate different peaks. A

Table 3: The number of peaks found (mean and standard deviation over 50 runs) by NN-FPSO with $I_{att} = 0.5$, $M = 30$, and different I_{sep} on MOPs

I_{sep}	F1	F2	F3	F4	F5(2D)	F5(3D)	F5(4D)
0	2.46±0.42	0.00±0.00	0.00±0.00	3.27±0.65	0.00±0.00	0.00±0.00	0.00±0.00
0.1	**5.00±0.00**	**4.00±0.00**	**2.00±0.00**	**5.00±0.00**	17.94±3.12	40.98±4.28	59.89±4.92
0.5	4.97±0.73	3.82±0.61	1.92±0.34	4.95±0.40	**18.00±0.00**	**41.24±4.23**	**60.04±4.21**
1	4.85±0.64	3.75±0.54	1.85±0.32	4.92±0.38	17.04±3.07	38.25±4.16	57.32±4.03
5	1.45±0.32	1.36±0.12	1.05±0.22	2.12±0.19	6.23±1.25	9.25±1.17	12.57±3.15
10	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00	0.00±0.00

larger value of I_{att} may drive NN-FPSO to converge onto some areas more quickly, that is, particles are rapidly trapped into some local optima. Hence, we recommend setting $I_{att} = 0.5$ in this study.

Figure 5(b) shows that on the Shubert 3D and Shubert 4D functions, the number of peaks found by NN-FPSO within 200000 evaluations is not much affected by I_{att} in the range $[0.1, 1.0]$. It can be observed that the best result on both test functions was achieved by setting I_{att} to around 0.5. Although NN-FPSO was unable to find all global optima on these functions, NN-FPSO is able to locate 42.95 global peaks on the average on Shubert 3D, showing that it has the search power of differentiating the peaks within each of the 27 groups of global peaks. On Shubert 4D, NN-FPSO is able to locate 62.5 global peaks on the average, but is unable to locate all of the 81 groups of global peaks in the search space.

5.3.3. Effect of the near-neighbor repulsion factor (I_{rep})

The near-neighbor repulsion mechanism is another key strategy introduced into NN-FPSO for MOPs. This mechanism is expected to prevent a particle from being attracted away from its local optimum by other fitter particles, and pushing it to move towards the global optimum in its "responsibility" under the guidance of the near-neighbor repulsion force. Similar to the previous test of the near-neighbor attraction mechanism, in order to examine the effect of the near-neighbor repulsion scheme on the behavior of NN-FPSO for different multimodal problems, experiments were carried out with $I_{att} = 0.5$ and I_{rep} set to 0, 0.1, 0.5, 1, 5, and 10, respectively. The experimental results regarding the number of peaks found by NN-FPSO within 200000 evaluations over 50 runs are provided in Table 3.

From Table 3, it can be seen that the optimal setting for I_{ref} is around

0.1 for $F1$-$F4$ and around 0.5 for $F5(2D)$, $F5(3D)$, and $F5(4D)$, respectively. This indicates that the near-neighbor repulsion scheme is important to maintain the diversity, which is a key consideration especially for locating multiple global optima. NN-FPSO performs the worst when I_{rep} is set to 0 or 10. When $I_{rep} = 0$, particles are likely to be trapped onto their nearby local optima, that is, the locating ability does depress due to the diversity loss. In the case of $I_{rep} = 10$, particles tend to focus on the information derived from the inferior ones, but neglecting the fitter ones. By doing this for each iteration, many misleading directions may be produced, leaving the entire swarm ill-adapted to the multimodal environment.

Furthermore, we are interested in investigating the range of I_{rep} in which NN-FPSO is robust on different problems. The performance of NN-FPSO was tested with I_{rep} in the range of $[0.1, 1.0]$. The experimental results regarding the NN-FPSO's ability of locating global optima with $I_{att} = 0.5$ and I_{rep} set in the range of $[0.1, 1.0]$ are provided in Figure 6. From Figure 6, it can be seen that I_{rep} within the range $[0.1, 1.0]$ does not have a large effect on the performance of NN-FPSO. Hence, we recommend setting I_{rep} to 0.1, which is expected to prevent particles from exploiting within the promising areas and ensure a sufficient exploration search power.

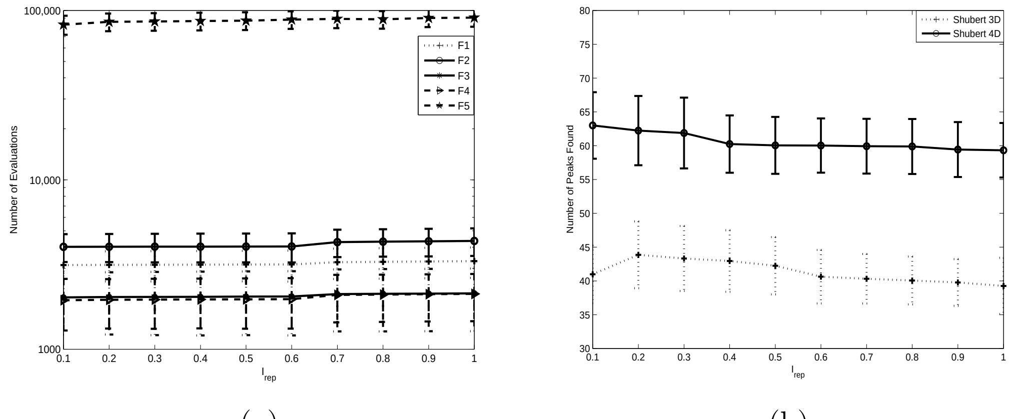


Figure 6: (a) The average number of evaluations needed to find all global optima on functions F_1-$F_5(2D)$ and (b) the average number of peaks found on Shubert 3D and Shubert 4D over 50 runs of NN-FPSO with $I_{att} = 0.5$ and different I_{ref}.

Table 4: Comparison results regarding the number of peaks found (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F1$-$F5$. The t-test results of comparing NN-FPSO with ARPSO algorithms are shown in brackets

MOP	ARSO	NN-FPSO
F1	4.72±0.05(+)	**5.00±0.00**
F2	3.86±0.35(+)	**4.00±0.00**
F3	1.98±0.45(+)	**2.00±0.00**
F4	4.96±0.63(+)	**5.00±0.00**
F5(2D)	16.89±0.75(+)	**18.00±0.00**

Table 5: Comparison results regarding the accuracy (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F1$-$F4$. The t-test results of comparing NN-FPSO with ARPSO algorithms are shown in brackets

MOP	ARPSO	NN-FPSO
F1	1.86E-2±5.62E-5(+)	**0.00E+0±0.00E+0**
F2	3.78E-1±7.55E+0(+)	**1.26E-5±1.28E-7**
F3	5.95E-3±5.34E-10(+)	**5.78E-9±7.21E-13**
F4	2.34E-3±3.24E-6(+)	**5.18E-6±8.75E-15**

5.4. Experimental comparison with ARPSO

This set of experiment provides the performance comparison of NN-FPSO and ARPSO on multimodal functions $F1$-$F5(2D)$. The experimental results regarding the number of peaks found and the accuracy are given in Table 4 and Table 5, respectively. In these tables, the statistical test results of comparing NN-FPSO with ARPSO by the one-tailed t-test with 98 degrees of freedom at a 0.05 level of significance are also given in the brackets, where the t-test result shown as "+" means that NN-FPSO is significantly better than ARPSO.

From Table 4 and Table 5, a significant result is that NN-FPSO outperforms ARPSO on all test functions. This result validates the efficiency of introducing the near-neighbor effect scheme into PSO for locating all global optima in parallel. The basic motivation behind NN-FPSO of allowing particles to search for different promising regions is beneficial for promoting the population diversity and searching for the suitable peak of each particle.

Table 6: Comparison results regarding the number of peaks found (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F1$-$F5$. The t-test results of comparing NN-FPSO with other PSO algorithms are shown in brackets

MOP	SPSO	FER-PSO	ANPSO	ESPSO	NN-FPSO
F1	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00**
F2	**4.00±0.00(∼)**	3.12±0.97(+)	9.24±0.39(+)	3.37±0.47(+)	**4.00±0.00**
F3	**2.00±0.00(∼)**	1.85±0.35(+)	1.92±0.42(+)	1.94±0.55(+)	**2.00±0.00**
F4	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00(∼)**	**5.00±0.00**
F5(2D)	16.81±0.17(+)	16.56±0.35(+)	17.54±0.65(∼)	16.87±0.42(+)	**18.00±0.00**

Table 7: Comparison results regarding the accuracy (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F1$-$F4$. The t-test results of comparing NN-FPSO with other PSO algorithms are shown in brackets

MOP	SPSO	FER-PSO	ANPSO	ESPSO	NN-FPSO
F1	1.98E-6±2.45E-7(+)	**0.00E+0±0.00E+0(∼)**	**0.00E+0±0.00E+0(∼)**	1.86E-6±3.27E-7(∼)	**0.00E+0±0.00E+0**
F2	9.78E-3±1.65E+0(+)	1.32E-4±6.79E-8(+)	1.37E-4±7.52E-6(+)	4.28E-4±3.68E-8(+)	**1.26E-5±1.28E-7**
F3	4.25E-5±4.45E-13(+)	6.24E-6±8.52E-9(+)	7.86E-6±6.32E-10(+)	8.67E-5±4.25E-10(+)	**5.78E-9±7.21E-13**
F4	1.86E-5±7.25E-9(+)	7.78E-5±9.86E-11(+)	5.42E-5±5.68E-10(+)	8.86E-5±8.24E-10(+)	**5.18E-6±8.75E-15**

5.5. Major experimental results and analysis

In this set of experiments, the performance of NN-FPSO in multimodal environments is compared with the four peer PSO algorithms, i.e., SPSO, FER-PSO, ANPSO, and ESPSO. These algorithms are presented for locating all global optima in parallel for MOPs. All corresponding parameters were fixed to be the same values for the multimodal test functions.

5.5.1. Comparison on MOPs in the low dimensional search space

This set of experiments investigates the performance comparison between the PSO algorithms on MOPs in the low dimensional search space, i.e., $F1$-$F4$. The experimental results regarding the number of peaks found, the accuracy, and the convergence speed, are given in Table 6, Table 7, and Table 8, respectively. In these tables, the statistical test results of comparing NN-FPSO with each peer PSO algorithm by the one-tailed t-test with 98 degrees of freedom at a 0.05 level of significance are also given in the brackets, where the t-test result is shown as "+", "−", or "∼" if NN-FPSO is significantly

Table 8: Comparison results regarding the convergence speed (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F1$-$F5(2D)$. The t-test results of comparing NN-FPSO with other PSO algorithms are shown in brackets

MOP	SPSO	FER-PSO	ANPSO	ESPSO	NN-FPSO
F1	3169±692(+)	4214±704(+)	5220±3323(+)	3720±704(+)	**3143±625**
F2	4069±731(+)	6788±821(+)	16308±13157(+)	4228±802(+)	**4023±753**
F3	2872±827(+)	3088±924(+)	2798±857(+)	2383±917(+)	**2022±804**
F4	2007±703(+)	2055±834(+)	6124±2465(+)	2013±837(+)	**1945±625**
F5(2D)	166050±42214(+)	188423±45677(+)	82248±10608(+)	272216±45272(+)	**82428±10575**

better than, significantly worse than, or statistically equivalent to a peer algorithm, respectively.

From Tables 6, 7, and 8, it can be seen that increasing the problem complexity poses a difficult issue for all PSO algorithms in locating all global optima in parallel. Another significant result is that NN-FPSO outperforms other PSO algorithms on all investigated cases. This validates the efficiency of introducing the force-imitated mechanism based on the near-neighbor effect into PSO for MOPs. The new movement strategy can integrate valuable information efficiently, and the combination of the near-neighbor attraction scheme with the near-neighbor repulsion scheme has an intensive exploration ability that helps particles search for global optima continuously rather than converging into local optima nearby.

5.5.2. Comparison on MOPs in the 3- and 4-dimensional search space

The functions $F5(3D)$, $F5(4D)$, and 3-dimensional $F6$, denoted $F6(3D)$, pose a serious challenge to niching algorithms that rely on a fixed niche radius parameter [40]. Based on above discussions, the swarm size was set to 1500 and the maximal number of evaluations was set to 200000 in this set of experiments. Even with a large swarm size, it is difficult for the PSO algorithms to find all peaks in one run. Hence, we measured the *number of peaks found* by each algorithm. The experimental results regarding the number of peaks found are given in Table 9.

Table 9 shows that NN-FPSO gives better performance than other PSO variants. The results suggest that it may be preferable to use the near-neighbor attraction and repulsion effects to locate each optimum, rather than a niching method relying on a fixed or adaptive niche radius value.

Table 9: Comparison results regarding the number of peaks found (mean and standard deviation over 50 runs) after 200000 evaluations of algorithms on $F5(3D)$, $F5(4D)$, and $F6(3D)$. The t-test results of comparing NN-FPSO with other PSO algorithms are shown in brackets

MOP	SPSO	FER-PSO	ANPSO	ESPSO	NN-FPSO
F5(3D)	8.47±0.32(+)	7.25±0.26(+)	8.26±0.75(+)	10.11±2.11(+)	**41.15±5.85**
F5(4D)	4.25±0.26(+)	3.11±0.31(+)	43.24±2.12(+)	45.67±2.57(+)	**61.42±6.73**
F6(3D)	75.92±2.46(+)	70.25±2.01(+)	76.25±1.97(+)	78.17±2.42(+)	**84.98±6.85**

Due to the near-neighbor identification process which integrates fitness and distance factors, NN-FPSO is able to develop stable "promising regions" on the majority of the global peaks.

5.5.3. Comparison on MOPs in the higher dimensional space

The Hump function $F7$ is a multimodal function generator that has been widely used as multimodal benchmark problems in the literature. The performance of the five PSO algorithms were also investigated on the Hump functions of 5 to 25 dimensions. In this study, the features of each peak i ($i = 1, 2, \cdots, K$) in the Hump functions were set as follows: the height $H_i = 1.0$ and the shape factor $\alpha_i = 1.0$. The radius of each peak is constant and increases with accordance to the number of dimensions ($R_i = 0.29$ for the 5-D Hump function instances, $R_i = 0.60$ for the 10-D instances, and $R_i = 1.45$ for the 25-D instances). The swarm size of PSO algorithms was set with the consideration of the complexity of functions. It was set to 800, 900, 1000, and 1100 for the 5-D Hump functions with 20, 30, 40, and 50 peaks, respectively. For the 10-D Hump functions with 20, 30, 40, and 50 peaks, the swarm size was set to 800, 900, 1000, and 1100, respectively. For the most difficult 25-D Hump functions with 20, 30, 40, and 50 peaks, the swarm size was set to 2700, 2800, 2900, and 3000, respectively.

The experimental results are provided in Table 10, where the t-test results of comparing NN-FPSO with other PSO algorithms with 98 degrees of freedom at a 0.05 level of significance are shown in brackets. From Table 10, it can be observed that NN-FPSO outperforms other algorithms in all test cases with different number of peaks. One major reason lies in that the two near-neighbor effects adopted in NN-FPSO continuously balance the abilities of exploiting known regions and exploring other optima in the search space.

Table 10: Comparison results regarding the number of peaks found (mean and standard deviation over 50 runs) by algorithms within 200000 evaluations on the Hump functions of different number of dimensions and different number of peaks, where K denotes the number of peaks in the search space. The t-test results of comparing NN-FPSO with other PSO algorithms are shown in brackets

K	SPSO	FER-PSO	ANPSO	ESPSO	NN-FPSO
			5-D Hump function		
20	19.82±0.06(∼)	17.85±0.72(+)	**20.00±0.00(∼)**	19.05±0.56(+)	**20.00±0.00**
30	29.61±0.32(∼)	28.73±0.87(+)	28.24±0.49(+)	28.37±0.68(+)	**30.00±0.00**
40	38.92±0.29(+)	36.54±0.92(+)	38.79±0.63(+)	39.12±0.79(+)	**40.00±0.00**
50	47.21±0.73(+)	48.56±0.62(+)	48.04±0.73(+)	48.97±0.52(+)	**49.75±0.85**
			10-D Hump function		
20	19.68±0.56(∼)	16.48±0.57(+)	18.32±0.49(+)	18.90±0.78(+)	**19.84±0.41**
30	26.46±1.47(+)	26.57±1.52(+)	27.31±0.52(+)	27.92±0.85(+)	**28.40±1.05**
40	36.21±2.42(+)	32.45±2.86(+)	36.63±0.32(+)	36.33±0.81(+)	**37.25±1.84**
50	46.98±4.16(+)	43.82±4.09(+)	46.53±0.73(+)	46.97±0.72(+)	**47.70±2.07**
			25-D Hump function		
20	4.24±0.07(+)	5.12±0.11(+)	8.63±0.23(+)	9.23±0.35(+)	**16.45±0.45**
30	6.46±0.12(+)	7.02±0.17(+)	9.24±0.39(+)	10.37±0.47(+)	**21.24±0.56**
40	18.92±0.15(+)	9.54±0.19(+)	10.32±0.45(+)	11.12±0.59(+)	**31.65±1.62**
50	9.81±0.17(+)	10.56±0.21(+)	12.54±0.53(+)	13.17±0.62(+)	**36.75±1.85**

6. Conclusion and future work

This paper presents a new PSO model, called near-neighbor effect based force-imitated PSO (NN-FPSO), with the combination of the near-neighbor effect in nature and the mechanics theory in physics for locating multiple global optima in parallel for multimodal optimization problems (MOPs). The basic motivation behind NN-FPSO is to utilize an efficiently integrated and interactive mechanism to simultaneously optimize particles toward multiple peaks. A new force-imitated equation is introduced into PSO, which produces a new version for PSO from the mechanics domain. This model uses the fitness and spatial information to identify the meaningful neighbors for each particle for a highly efficient interaction. Besides the personal best position, the movement of each particle is also guided by the following two forces:

- The near-neighbor attraction force. It is important to enable particles to identify promising areas where the optima may reside. One straightforward method is to allow particles nearby each other to work together toward their closest peak, rather than the fittest peak. Hence, we adopt the superior-and-nearer neighbor in the personal best sets to be an attraction force, and use the force-imitated mechanism presented in this study to improve each particle's movement to its possible peaks. It ensures an exploitation ability for promising regions, thereby driving particles to their local vicinity in the multimodal fitness landscape.

- The near-neighbor repulsion force. Although particles should investigate local promising regions, they need to ultimately progress toward the global optima. The near-neighbor repulsion force produced by the inferior-and-nearer neighbor prevents the collision of particles within a local region discovered and guarantees the swarm diversity to seek for other optima undiscovered, as well as exploring potentially more promising regions.

In order to justify the proposed NN-FPSO, experiments were carried out to compare the performance of NN-FPSO with a number of state-of-the-art PSO algorithms on a series of multimodal benchmark problems. From the experimental results, some conclusions can be drawn on the multimodal test problems. First, the introduction of the near-neighbor effect with the force-limited mechanism is beneficial for the performance of PSO in multimodal environments. Second, the proposed near-neighbor attraction scheme is efficient to improve the performance of NN-FPSO for locating all global optima in parallel. Third, the strategy of extracting information from other particles besides the best solutions as in the classic PSO algorithm is a good choice for preserving valuable information and avoiding convergence to improve the performance of PSO for multimodal problems.

Generally speaking, the experimental results indicate that the proposed NN-PSO algorithm can be a good optimizer for MOPs.

For future work, it would be valuable to improve the performance of NN-FPSO for more challenging MOPs, such as the Shubert 3D and Shubert 4D functions. In this study, the value of I_{att} and I_{rep} are fixed during the whole optimization process. It may be beneficial to adaptively adjust these two parameters according to some feedback information from the swarm, such as the population statistics at each iteration. Furthermore, the concept of the near-neighbor force in the study is similar to the attractive and repulsive force in mobile robot motion planning based on the potential field [43]. Therefore, it is also worthy to extend the force-imitated mechanism for more practical models, such as orienteering problems [10], dynamic multimodal problems, and dynamic multi-objective problems [37, 42], which also have wide practical applications in the real world.

Acknowledgement

The authors would like to thank the guest editors and anonymous reviewers for their thoughtful suggestions and constructive comments. This work was supported in part by the Key Program of National Natural Science Foundation (NNSF) of China under Grant 70931001, Grant 70771021, and Grant 70721001, the National Natural Science Foundation (NNSF) of China for Youth under Grant 61004121, Grant 70771021, the Science Fund for Creative Research Group of NNSF of China under Grant 60821063, the PhD Programs Foundation of Ministry of Education of China under Grant 200801450008, and in part by the Engineering and Physical Sciences Research Council (EPSRC) of UK under Grant EP/E060722/1 and Grant EP/E060722/2.

References

[1] D. Beasley, D. Bull, and R. Martin. A sequential niche technique for multimodal function optimization. Evol. Comput., 1(2): 101–125, 1993.

[2] S. Bird and X. Li. Adaptively choosing niching parameters in a PSO. In: Proc. of the 2006 Genetic and Evol. Comput. Conf., pp. 3–10, 2006.

[3] S. Bird and X. Li. Enhancing the robustness of a speciation-based PSO. In: Proc. of the 2006 IEEE Congress on Evol. Comput., pp. 843–850, 2006.

[4] T. M. Blackwell. Swarms in dynamic environments. In: Proc. of Genetic and Evolutionary Computation Conf (GECCO 2003), pp. 1–12, 2003.

[5] T. M. Blackwell and J. Branke. Multi-swarm optimization in dynamic environments. In: EvoWorkshops 2004: Applications of Evolutionary Computing, LNCS 3005, pp. 489–500, 2004.

[6] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. In: Advances in evolutionary computing: theory and applications, 2003.

[7] D. Bratton and T. Blackwell. Understanding particle swarms through simplification: a study of recombinant PSO. In: Proc. of the 2007 GECCO Conf. Companion on Genetic and Evol. Comput., pp. 2621–2628, 2007.

[8] R. Brits, A. Engelbrecht, and F. van den Bergh. A niching particle swarm optimizer. In: Proc. of the 4th Asia-Pacific Conf. on Simulated Evolution and Learning, vol. 2, pp. 692–696, 2002.

[9] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. on Evol. Comput., 2(1): 58–73, 2002.

[10] H. Dallard, S.S. Lam, S. Kulturel-Konak. Solving the orienteering problem using attractive and repulsive particle swarm optimization. In: Proc. of the 2007 IEEE Int. Conf. on Information Reuse and Integration, pp. 12–17, 2007.

[11] K. Deb and D. Goldberg. An investigation of niche and species formation in genetic function optimization. In: Proc. of the 3rd Int. Conf. on Genetic Algorithms, pp. 42–50, 1989.

[12] K. De Jong. An analysis of the behavior of a class of genetic adaptive systems. PhD Dissertation, University of Michigan, 1975.

[13] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In: Proc. of the 6th Int. Symp. on Micro Machine and Human Science, pp. 39–43, 1995.

[14] D. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In: Proc. of the 2nd Int. Conf. on Genetic Algorithms, pp. 41–49, 1987.

[15] G. R. Harik. Finding multimodal solutions using restricted tournament selection. In: Proc. of the 6th Int. Conf. on Genetic Algorithms, pp. 24–31, 1995.

[16] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In: Proc. of the 1995 IEEE Int. Conf. on Neural Networks, pp. 1942–1948, 1995.

[17] J. Kennedy and R. Eberhart. Swarm Intelligence. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001.

[18] J. Kennedy. In search of the essential particle swarm. In: Proc. of the 2006 IEEE Congress on Evol. Comput., pp. 6158–6165, 2006.

[19] J. Li, M. Balazs, G. Parks, and P. Clarkson. A species conserving genetic algorithm for multimodal function optimization. Evol. Comput., 10(3): 207–234, 2002.

[20] X. Li. Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Proc. of the 2004 Genetic and Evol. Comput. Conf. (GECCO'04), LNCS 3102, pp. 105–116, 2004.

[21] X. Li. Efficient differential evolution using speciation for multimodal function optimization. In: Proc. of the 2005 Conf. on Genetic and Evol. Comput., pp. 873–880, 2005.

[22] X. Li. A multimodal particle swarm optimizer based on fitness euclidean-distance ratio. In: Proc. of the 9th Annual Conf. on Genetic and Evol. Comput., pp. 78–85, 2007.

[23] S. Mahfoud. Simple analytical models of genetic algorithms for multimodal function optimization. Illinois Genetic Algorithm Laboratory Technical Report No. 94005, Department of Computer Science, University of Lllinois at Urbana-Champaign, Urbana, IL, USA, 1993.

[24] S. Mahfoud. Niching method for genetic algorithms. PhD Dissertation. Illinois Genetic Algorithms Laboratory (IlliGAL) Technical Report No. 95001, Department of Computer Science, University of Lllinois at Urbana-Champaign, Urbana, IL, USA, 1995.

[25] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: simpler, maybe better. IEEE Trans. Evol. Comput., 8(3): 204–210, June 2004.

[26] R. Mendes and J. Neves. What makes a successful society? experiments with population topologies in particle swarms. In: Proc. of the 17th Brazilian Symposium on Artificial Intelligence (SBIA 2004), LNCS 3171, pp. 346–355, 2004.

[27] O. Mengsheol and D. Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. In: Proc. of the 1999 Genetic and Evol. Comput. Conf. (GECCO-99), pp. 409–416, 1999.

[28] D. Parrott and X.Li. A particle swarm model for tacking multiple peaks in a dynamic environment using speciation. In: Proc. of the 2004 Genetic and Evol. Comput. Conf., vol. 1, pp. 98–103, 2004.

[29] K. E. Parsopoulos and M.N. Vrahitis. Modification of the particle swarm optimizer for locating all the global minima. In: Proc. of the Int. Conf. on Artificial Neural Networks and Genetic Algorithms (ICANNGA 2001), pp. 324-327, 2001.

[30] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In: Proc. of the 3rd IEEE Int. Conf. on Evol. Comput., pp. 798–803, 1996.

[31] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization: an overview. *Swarm Intelligence* , 1(1): 33–58, 2007.

[32] A. Rajac, J. Wongsriratanakul, and S. Rajac. Magnetic ordering in an organic polymer. *Science*, 294: 1503–1505, 2001.

[33] E. Rashedi, H. Nezamabadi-pour, and Saryazdi S. GSA: A gravitational search algorithm. *Information Sciences*, 179(13): 2232-2248, June 2009.

[34] J. Riget and J. S. Vesterstroem. A diversity-guided particle swarm optimizer - the ARPSO, EVALife Technical Report No. 2002-02, 2002.

[35] G. Singh and K. Deb. Comparison of multi-modal optimization algorithms based on evolutionary algorithms. In: Proc. of the 2006 Genetic and Evol. Comput. Conf., pp. 1305–1312, 2006.

[36] I. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85(6): 317–325, 2003.

[37] P. K. Tripathi, S. Bandyopadhyay and S. K. Pal. Multi-objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, 177(22): 5033-5049, November 2007.

[38] T. M. Van Dam and T. A. Herring. Detection of atmospheric pressure loading using very long baseline interferometry measurements. *J. Geophys. Res*, 99(B3): 4505–4517, 1994.

[39] F. van den Bergh. An Analysis of Particle Swarm Optimizers, PhD Thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.

[40] F. van den Bergh and A. P. Engelbrecht. Study of particle swarm optimization particle trajectories. *Information Sciences*, 176(8): 937–971, 2006.

[41] K. Veeramachaneni, T. Peram, C. Mohan, and L. Osadciw. Optimization using particle swarm with near neighbor interactions. In: Proc. of the 2003 Genetic and Evol. Comput. Conf., pp. 110–121, 2003.

[42] Y. J. Wang and Y. P. Yang. Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, 179(12): 1944-1959, May 2009.

[43] Q. D. Zhu, Y. J. Yan, and Z. Y. Xing. Robot path planning based on artificial potential field approach with simulated annealing, In: Proc. of the 6th Int. Conf. on Intelligent Systems Design and Applications, vol. 2, pp. 622-627, 2006.