Developing Niching Algorithms in Particle Swarm Optimization

Xiaodong Li

School of Computer Science and Information Technology, RMIT University, Melbourne, Australia
xiaodong.li@rmit.edu.au

Abstract. Niching as an important technique for multimodal optimization has been used widely in the Evolutionary Computation research community. This chapter aims to provide a survey of some recent efforts in developing state-of-the-art PSO niching algorithms. The chapter first discusses some common issues and difficulties faced when using niching methods, then describe several existing PSO niching algorithms and how they combat these problems by taking advantages of the unique characteristics of PSO. This chapter will also describe a recently proposed *lbest* ring topology based niching PSO. Our experimental results suggest that this *lbest* niching PSO compares favourably against some existing PSO niching algorithms.

1 Introduction

Stochastic optimization algorithms such as Evolutionary Algorithms (EAs) and more recently Particle Swarm Optimization (PSO) algorithms have shown to be effective and robust optimization methods for solving difficult optimization problems. The original and many existing forms of EAs and PSOs are usually designed for locating a single global solution. These algorithms typically converge to one final solution because of the global selection scheme used. However, many real-world problems are "multimodal" by nature, that is, multiple satisfactory solutions exist. For such an optimization problem, it might be desirable to locate all global optima and/or some local optima that are also considered as being satisfactory. Numerous techniques have been developed in the past for locating multiple optima (global or local). These techniques are commonly referred to as "niching" methods. A niching method can be incorporated into a standard EA to promote and maintain the formation of multiple stable subpopulations within a single population, with an aim to locate multiple optimal or suboptimal solutions. Niching methods are of great value even when the objective is to locate a single global optimum. Since a niching EA searches for multiple optima in parallel, the probability of getting trapped on a local optimum is reduced.

Niching methods have also been incorporated into PSO algorithms to enhance their ability to handle multimodal optimization problems. This chapter aims to provide a survey of several state-of-the-art PSO niching algorithms. The chapter will begin with a brief background on niching methods in general, and then identify some difficulties faced by existing niching methods. The chapter will then go on to describe the development of several PSO niching algorithms and how they are designed to resolve some of these issues by taking advantages of the inherent characteristics of PSO. In particular, the chapter will describe in detail a recently proposed *lbest* ring topology based niching PSO. Experimental results on this *lbest* niching PSO will be compared against an existing PSO niching algorithm, and their strengthes and weaknesses will be examined. Finally the chapter concludes by summing up the important lessons learnt on developing competent PSO niching methods, and possible future research directions.

2 Niching Methods

Just like Evolutionary Algorithms themselves, the notion of niching is inspired by nature. In natural ecosystems, individual species must compete to survive by taking on different roles. Different species evolve to fill different *niches* (or subspaces) in the environment that can support different types of life. Instead of evolving a single population of individuals indifferently, natural ecosystems evolve different species (or subpopulations) to fill different niches. The terms species and niche are sometimes interchangeable. Niching methods were introduced to EAs to allow maintenance of a population of diverse individuals so that multiple optima within a single population can be located [25]. One of the early niching methods was developed by De Jong in a scheme called *crowding*. In *crowding*, an offspring is compared to a small random sample taken from the current population, and the most similar individual in the sample is replaced. A parameter *CF* (*crowding factor*) is commonly used to determine the size of the sample. The most widely used niching method is probably *fitness sharing*. The sharing concept was originally introduced by Holland [16], and then adopted by Goldberg and Richardson [14] as a mechanism to divide the population into different subpopulations according to the similarity of the individuals in the population. Fitness sharing was inspired by the *sharing* concept observed in nature, where an individual has only limited resources that must be shared with other individuals occupying the same niche in the environment. A sharing function is often used to degrade an individual's fitness based on the presence of other neighbouring individuals. Although *fitness sharing* has proven to be a useful niching method, it has been shown that there is no easy task to set a proper value for the sharing radius parameter σ_{share} in the sharing function without prior knowledge of the problems [13].

Apart from the above, many more niching methods have been developed over the years, including *derating* [1], *deterministic crowding* [24], *restricted tournament selection* [15], *parallelization* [2], *clustering* [37], and *speciation* [30, 20]. Niching methods have also been developed for PSO, such as *NichePSO* [31], *SPSO* [26], and *VPSO* [33].

2.1 Difficulties Facing Niching Methods

Most of these niching methods, however, have difficulties which need to be overcome before they can be applied successfully to real-world multimodal problems. Some identified issues include the following:

- Reliance on prior knowledge of some niching parameters, which must be set with some optimal values so that the optimization algorithm can perform well. A common use of a niching parameter is to tell how far apart two closest optima are. A classic example is the sharing parameter σ_{share} in fitness sharing [14]. Other uses of niching parameters include *crowding factor* in *crowding method* [12], the window size w in *restricted tournament selection* [15], or the number of clusters in *k-means clustering methods* [37, 17].
- Difficulty in maintaining found solutions in a run. Some found solutions might be lost in successive generations. For example, the original De Jong's crowding has been shown unable to maintain all found peaks during a run [24]. A good niching algorithm should be able to form and maintain stable subpopulations over the run.
- In traditional niching EAs, it was observed that crossover between two fit individuals from different niches could produce far less fit offspring than the parents [25]. How can we minimize such detrimental crossover operations across different niches?
- Some existing niching methods are designed only for locating all global optima, while ignoring local optima. Examples include the sequential niche GA (SNGA) [1], clearing [30], SCGA [20], NichePSO [31], and SPSO [21, 26]. However, it might be desirable to obtain both global and local optima in a single run.
- Most niching methods are evaluated on test functions of only 2 or 3 dimensions. How well these niching algorithms perform on high dimensional problems remain unclear.
- Higher computational complexity. Most of the niching algorithms use global information calculated from the entire population, therefore require at least $\mathcal{O}(N^2)$ computational complexity (where N is the population size). Many niching algorithms suffer from this problem.
- Most existing niching methods are evaluated using static functions. When functions can vary over time, ie., the multimodal fitness landscape may change over time, most existing niching methods are unable to cope with the dynamically changing environments.

Problems with Niching Parameters

Most existing niching methods, however, suffer from a serious problem - their performance is subjected heavily to some niching parameters, which are often difficult to set by a user. For example the sharing parameter σ_{share} in *fitness sharing* [14], the species distance σ_s in *species conserving GA* (SCGA) [20], the distance measure σ_{clear} in *clearing* [30], and the species radius r_s in the

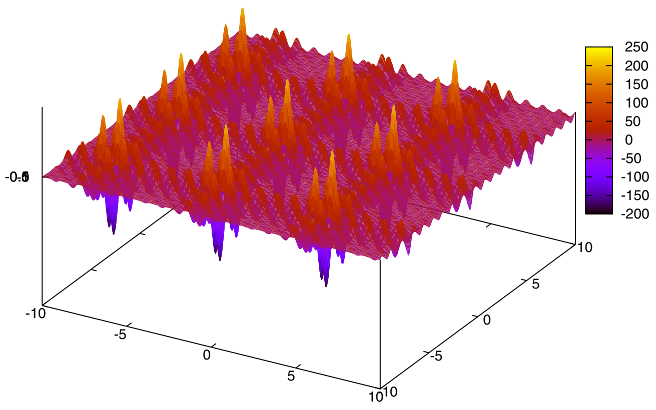


Fig. 1. Inverted Shubert 2D function.

speciation-based PSO (SPSO) [26]. Sometimes niching parameters can be under different disguises, such as the *crowding factor* in *crowding* [12], the window size w in *restricted tournament selection* [15], or the number of clusters in k-*means clustering methods* [37, 17]. The performance of these EAs depend very much on how these parameters are specified. Unfortunately, in many real-world problems such prior knowledge are often unavailable. Some recent works by Bird and Li [4, 5] attempted to reduce the sensitivity of the SPSO to the niche radius parameter values. However, either this parameter still remains (though made more robust), or several new parameters are introduced. It would be desirable if a user can be completely freed from specifying any niching parameters.

Fig. 1 shows an example of a function fitness landscape that has 9 pairs of global optima and numerous local optima. Within each pair, two global optima are very close to each other but optima from different pairs are further away. A niching algorithm relying on a fixed niche radius value to determine a particle's membership in a niche would have a significant difficulty to work properly on such a landscape. To capture all peaks, a niching EA would have to set its niche radius extremely small so that the closest two peaks can be distinguished. However, doing so would form too many small niches, with possibly too few individuals in each niche. As a result, these niches tend to prematurely converge. On the other hand, if the niche radius is set too large, peaks with a distance between them smaller than this value will not be distinguished. In short, it is likely that there is no optimal value for the niche radius parameter. Dependency on a fixed niche radius is a major drawback for niching methods that rely on such a parameter. For example in [20], on the inverted Shubert 2D function (as shown in Fig. 1), SCGA had to be tuned with a radius value of 0.98 and a population size of 1000 in order to locate all 18 global peaks reliably [20].

For Shubert 3D, SCGA used a population size of 4000 in order to locate all 81 global peaks. As dimension increased to 4, SCGA was only able to identify groups of global peaks, but not individual global optima within each group. Another similar niching algorithm SPSO [26] suffers the same problem.

3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a Swarm Intelligence technique originally developed from studies of social behaviours of animals or insects, e.g., bird flocking or fish schooling [18]. In a canonical PSO, the velocity of each particle is modified iteratively by its *personal best* position (i.e., the position giving the best fitness value so far), and the position of best particle from the entire swarm. As a result, each particle searches around a region defined by its personal best position and the position of the population best. Let's use $\mathbf{v}_i$ to denote the velocity of the i-th particle in the swarm, $\mathbf{x}_i$ its position, $\mathbf{p}_i$ the best position it has found so far, and $\mathbf{p}_g$ the best position found from the entire swarm (so called global best). $\mathbf{v}_i$ and $\mathbf{x}_i$ of the i-th particle in the swarm are updated according to the following two equations [10]:

$$\mathbf{v}_i \leftarrow \chi(\mathbf{v}_i + \mathbf{R}_1[0,\varphi_1] \otimes (\mathbf{p}_i - \mathbf{x}_i) + \mathbf{R}_2[0,\varphi_2] \otimes (\mathbf{p}_g - \mathbf{x}_i)), \quad (1)$$

$$\mathbf{x}_i \leftarrow \mathbf{x}_i + \mathbf{v}_i, \quad (2)$$

where $\mathbf{R}_1[0,\varphi_1]$ and $\mathbf{R}_2[0,\varphi_2]$ are two separate functions each returning a vector comprising random values uniformly generated in the range $[0, \varphi_1]$ and $[0, \varphi_2]$ respectively. φ_1 and φ_2 are commonly set to $\frac{\varphi}{2}$ (where φ is a positive constant). The symbol $\otimes$ denotes point-wise vector multiplication. A constriction coefficient χ is used to prevent each particle from exploring too far away in the search space, since χ applies a dampening effect to the oscillation size of a particle over time. This Type 1" constricted PSO suggested by Clerc and Kennedy is often used with χ set to 0.7298, calculated according to $\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$, where $\varphi = \varphi_1 + \varphi_2 = 4.1$ [10].

3.1 PSO Niching Methods

This section describes several representative niching methods that have been developed in conjunction with PSO.

Stretching Method

In [27], Parsopoulos and Vrahitis introduced a method in which a potentially good solution is isolated once it is found, then the fitness landscape is 'stretched' to keep other particles away from this area of the search space [28], similar to the derating method used in SNGA [1]. The isolated particle is checked to see if it is a global optimum, and if it is below the desired accuracy, a small population is generated around this particle to allow a finer search in this area.

The main swarm continues its search in the rest of the search space for other potential global optima. With this modification, Parsopoulos and Vrahitis' PSO was able to locate all the global optima of some selected test functions successfully. However, the drawback is that this *stretching* method introduces several new parameters which are difficult to specify in the *stretching* function, as well as the risk of introducing false optima as a result of *stretching*.

NichePSO

Brits *et al.* proposed NichePSO [31], which further extended Parsopoulos and Vrahitis's model. In NichePSO, multiple subswarms are produced from a main swarm population to locate multiple optimal solutions in the search space. Subswarms can merge together, or absorb particles from the main swarm. NichePSO monitors the fitness of a particle by tracking its variance over a number of iterations. If there is little change in a particle's fitness over a number of iterations, a subswarm is created with the particles closest neighbor. The issue of specifying several user parameters still remains. The authors also proposed *nbest* PSO in [9], where a particle's neighbourhood best is defined as the average of the positions of all particles in its neghbourhood. By computing the Euclidean distances between particles, the neighbourhood of a particle can be defined by its k closest particles, where k is a user-specified parameter. Obviously the performance of *nbest* PSO depends on how this parameter is specified.

Speciation-Based PSO

The speciation-based PSO (SPSO) model was developed based on the notion of species [21]. The definition of species depends on a parameter r_s, which denotes the radius measured in Euclidean distance from the center of a species to its boundary. The center of a species, the so-called species seed, is always the best-fit individual in the species. All particles that fall within the r_s distance from the species seed are classified as the same species.

The procedure for determining species seeds, introduced by Pétrowski in [30] and also Li et al. in [20], is adopted here. By applying this algorithm at each iteration step, different species seeds can be identified for multiple species and these seeds' $\mathbf{p_i}$ can be used as the $\mathbf{p_g}$ (like a neighbourhood best in a *lbest* PSO) for different species accordingly. Algorithm 1 summarizes the steps for determining species seeds.

Algorithm 1 is performed at each iteration step. The algorithm takes as an input L_{sorted}, a list containing all particles sorted in decreasing order of their $\mathbf{x_i}$ fitness. The species seed set S is initially set to $\emptyset$. All particles' $\mathbf{x_i}$ are checked in turn (from best to least-fit) against the species seeds found so far. If a particle does not fall within the radius r_s of all the seeds of S, then this particle will become a new seed and be added to S. Fig. 2 provides an example to illustrate the working of this algorithm. In this case, applying the algorithm will identify s_1, s_2 and s_3 as the species seeds. Note also that if seeds have their radii overlapped (e.g., s_2 and s_3 here), the first identified seed (such as s_2) will dominate over

```
input  : $L_{sorted}$ - a list of all particles sorted in their decreasing $f(\mathbf{x_i})$ values
output: $S$ - a list of all dominating particles identified as species seeds
begin
    $S = \emptyset$;
    while not reaching the end of $L_{sorted}$ do
        Get best unprocessed $p \in L_{sorted}$;
        $found \leftarrow$ FALSE;
        for all $s \in S$ do
            if $d(s,p) \leq r_s$ then
                $found \leftarrow$ TRUE;
                break;
        if not $found$ then
            let $S \leftarrow S \cup \{p\}$;
end
```

Algorithm 1. The algorithm for determining species seeds according to all $f(\mathbf{x_i})$ values.

those seeds identified later from the list L_{sorted}. For example, s_2 dominates s_3 therefore p should belong to the species led by s_2.

Since a species seed is the best-fit particle's $\mathbf{x_i}$ within a species, other particles within the same species can be made to follow the species seed's $\mathbf{p_i}$ as the newly identified neighborhood best. This allows particles within the same species to be attracted to positions that make them even fitter. Because species are formed around different optima in parallel, making species seeds the new neighborhood bests provides the right guidance for particles in different species to locate multiple optima.

Since species seeds in S are sorted in the order of decreasing fitness, the more highly fit seeds also have a potentially larger influence than the less fit seeds. This also helps the algorithm to locate the global optima before local ones.

Once the species seeds have been identified from the population, we can then allocate each seed's $\mathbf{p_i}$ to be the $\mathbf{p_g}$ to all the particles in the same species at each iteration step. The speciation-based PSO (SPSO) accommodating the algorithm for determining species seeds described above can be summarized in Algorithm 2.

In SPSO, a niche radius must be specified in order to define the size of a niche (or species). Since this knowledge might not be always available a priori, it might be difficult to apply this algorithm to some real-world problems. To combat this problem, two extensions to SPSO aiming to improve the robustness to such a niching parameter were proposed in [4, 5]. In [4], population statistics were used to adaptively determine the niching parameters during a run (see also section 3.1), whereas in [5], a time-based convergence measure was used to directly enhance SPSO's robustness to the niche radius value. These extensions to SPSO made it more robust. Nevertheless, the need to specify the niche radius still remains.

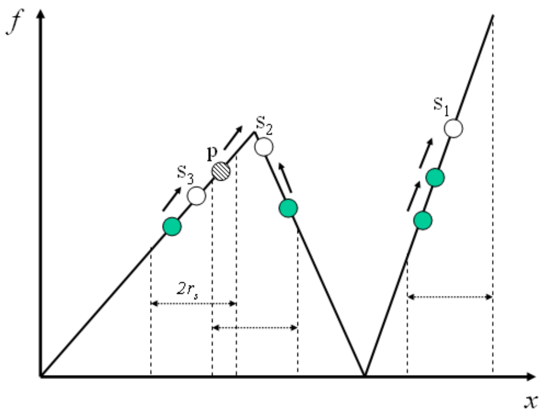


Fig. 2. An example of how to determine the species seeds from a population of particles. s_1, s_2 and s_3 are chosen as the species seeds. Note that p follows s_2.

```
//initialization;
for i=1 to popSize do
    randomly initialize i-th particle: $\mathbf{v_i}, \mathbf{x_i}$;
    $\mathbf{p_i} \leftarrow \mathbf{x_i}$
repeat
    for i=1 to popSize do
        evaluate $f(\mathbf{x_i})$;
        if $f(\mathbf{x_i}) > f(\mathbf{p_i})$ then
            $\mathbf{p_i} \leftarrow \mathbf{x_i}$
    Sort all particles according to their fitness values (from the best-fit to the least-fit);
    Call the speciation procedure in Algorithm 1 to identify species seeds;
    Assign each identified species seed's $\mathbf{p_i}$ as the $\mathbf{p_g}$ to all individuals identified in the same species;
    Adjust particle positions using PSO update equations (1) and (2);
    Check each species to see if the numParticles $> p_{max}$; If so, replace the excess particles with random particles into the search space;
until the termination condition is met ;
```

Algorithm 2. The species based PSO algorithm.

Adaptive Niching PSO

Instead of requiring a user to specify the niche radius r, the Adaptive Niching PSO (ANPSO) proposed in [4, 3] adaptively determines it from the population statistics at each iteration. More specifically, ANPSO sets r to the average distance between every particle and its closest neighbour (see Fig. 3), as follows:

$$r = \frac{\sum_{i=1}^{N} min_{j \neq i} ||x_i - x_j||}{N}. \tag{3}$$

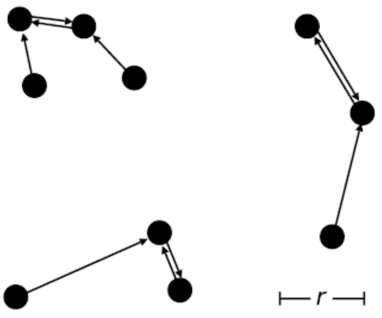


Fig. 3. Calculating the distance from each particle to the particle closest to it. r is calculated by averaging these distances.

An undirected graph g is then created containing a node for each particle. If ANPSO finds pairs of particles that are within r of each other for several iterations, a niche is formed. The remaining unconnected particles (ie., unniched) are mapped onto a von Neumann neighbourhood. At each iterations, particles can join or be removed from existing niches. Whereas the standard PSO updates are applied to particles in each niche, the *lbest* PSO according to the von Neumann neighbourhood topology is used to update particles that are classified as unniched. The unniched particles are useful especially to search more broadly around the problem space. ANPSO removes the need to specify niche radius r in advance, however, at the same time, it introduces two new parameters, the number of steps two particles must be close before forming a niche, and the maximum number of particle in each niche. Nevertheless, at least on some multimodal test functions, ANPSO's performance was shown to be less sensitive to these two parameters.

Fitness-Euclidean Distance Ratio Based PSO

A PSO based on Fitness-Euclidean distance Ratio (FER-PSO) was proposed in [22]. In FER-PSO, personal bests of the particles are used to form a *memory-swarm* to provide a stable network retaining the best points found so far by the population, while the current positions of particles act as parts of an *explorer-swarm* to explore broadly around the search space. Instead of using a single global best, each particle is attracted towards a fittest-and-closest neighbourhood point $\mathbf{p}_n$ which is identified via computing its FER (Fitness and Euclidean-distance Ratio) value:

$$FER_{(j,i)} = \alpha \cdot \frac{f(\mathbf{p}_j) - f(\mathbf{p}_i)}{||\mathbf{p}_j - \mathbf{p}_i||}, \tag{4}$$

where $\alpha = \frac{||s||}{f(\mathbf{p}_g) - f(\mathbf{p}_w)}$ is a scaling factor, to ensure that neither fitness nor Euclidean distance becomes too dominated over one another. $||s||$ is the size of the

```
input  : A list of all particles in the population
output: Neighbourhood best $\mathbf{p}_n$ based on the $i$-th particle's FER value
$FER \leftarrow 0$, $tmp \leftarrow 0$, $euDist \leftarrow 0$ ;
for $j = 1$ to Population Size do
    Calculate the Euclidean distance euDist from $\mathbf{p}_i$ to the $j$-th particle's
    personal best $\mathbf{p}_j$;
    if (euDist not equal to 0) then
        Calculate $FER$ according to equation (4) ;
        if (j equal to 1) then $tmp \leftarrow FER$;
        if ($FER > tmp$) then
            $tmp \leftarrow FER$ ;
            $\mathbf{p}_n \leftarrow \mathbf{p}_j$ ;
return $\mathbf{p}_n$
```

Algorithm 3. The pseudocode of calculating $\mathbf{p}_n$ for the i-th particle under consideration, according to its FER value. To obtain $\mathbf{p}_n$ for all particles, this algorithm needs to be iterated over the population.

search space, which can be estimated by its diagonal distance $\sqrt{\sum_{k=1}^{Dim}(x_k^u - x_k^l)^2}$ (where x_k^u and x_k^l are the upper and lower bounds of the k-th dimension of the search space). $\mathbf{p}_w$ is the worst-fit particle in the current population.

FER-PSO is able to reliably locate all global optima, given that the population size is sufficiently large. One noticeable advantage is that FER-PSO does not require specification of niching parameters. Nevertheless, it introduces a parameter α which needs to be determined by the upper and lower bounds of the variables. Since the algorithm uses global information, the complexity of the algorithm is $\mathcal{O}(N^2)$ (where N is the population size).

Vector-Based PSO

In [34, 33], a vector-based PSO (VPSO) was developed by treating each particle as a vector and simply carrying out vector operations over them. For each particle, VPSO computes the dot product Δ of two differential vectors, $\mathbf{p_i} - \mathbf{x_i}$ and $\mathbf{p_i} - \mathbf{x_i}$. A niche is defined by a niche radius determined by the distance between $\mathbf{p_g}$ and the nearest particle with a negative dot product (ie., moving in an opposite direction). Niche identification is done in a sequential manner. Once a niche is determined, it is excluded from the population, and the process is repeated on the remaining population, until the entire population is grouped into various niches. In VPSO it is not required to specify the niche radius parameter. However, the distance calculations can be expensive since every particle has to be compared with all remaining particles in the population.

In a subsequent work [35], PVPSO which is a parallel version of VPSO was proposed. In PVPSO, different niches can be maintained in parallel. A special procedure was also introduced to merge niches if they are too close to each other (below a specified threshold ϵ).

Clustering-Based PSO

The use of clustering techniques for PSO was first proposed by Kennedy in [17], where the k-means clustering algorithm was used to identify the centers of different clusters of particles in the population, and then use these cluster centers to substitute the personal bests or neighborhood bests. However, Kennedy's clustering technique was used to help locate a single global optimum, rather than multiple optima, as niching normally does. Inspired by this work, a k-means clustering PSO (kPSO) for niching was proposed in [29]. In kPSO, k-means is repeatedly applied to the swarm population at a regular interval. Between each interval, PSO is executed in the normal manner. Particles in different clusters at an early stage of a run could end up in the same cluster as they converge towards the same local optimum. The parameter k is estimated by using the Bayesian information criterion (BIC) [36]. More specifically, k-means is repeatedly applied to the population with different k values (usually from 2 to $N/2$), and the resulting clustering that has the highest BIC value is chosen. By doing this, there is no need to specify k in kPSO. It was shown that the performance of kPSO was comparable to existing PSO niching algorithms such as SPSO and ANPSO on some multimodal test functions.

Niching PSOs for Dynamically Changing Multimodal Environments

Many real-world optimization problems are dynamic and require optimization algorithms capable of adapting to the changing optima over time. An environment that is both multimodal and dynamic presents additional challenges. In fully dynamic multimodal environments, optima may shift spatially, change both height and shape or come into or go out of existence. One useful approach in handling this is to divide the population into several subpopulations, with each subpopulation searches for a promising region of the search space simultaneously. This is the core idea of several recently proposed PSO niching algorithms for handling a dynamical multimodal landscape such as the Dynamic SPSO [26] and the multi-swarm PSO (MPSO) [8], and rSPSO [6]. Several additional issues must be addressed, including outdated memory, population re-diversification, change detections and response strategies. For further information, readers can refer to [7].

4 New Niching Methods Using a *lbest* PSO

In [23], a novel PSO niching method was developed using a simple ring neighbourhood topology, which belongs to the class so called *lbest* PSO models. This

PSO niching method makes use of the inherent characteristics of PSO and does not require any niching parameters. It can operate as a niching algorithm by using individual particles' local memories to form a stable network retaining the best positions found so far, while these particles explore the search space more broadly. Given a reasonably large population uniformly distributed in the search space, the ring topology based niching PSOs are able to form stable niches across different local neighbourhoods, eventually locating multiple global/local optima. This section describes several such ring topology based niching PSO variants in detail, and how PSO's inherent characteristics such as *memory-swarm* and *explorer-swarm* can be utilized to induce stable niching behaviours.

Generally speaking, two common approaches of choosing $\mathbf{p}_g$ in equation (1) are known as *gbest* and *lbest* methods. In a *gbest* PSO, the position of each particle in the search space is influenced by the best-fit particle in the entire population, whereas a *lbest* PSO only allows each particle to be influenced by the best-fit particle chosen from its neighborhood. The *lbest* PSO with a neighborhood size set to the population size is equivalent to a *gbest* PSO. Kennedy and Mendes [19] studied PSOs with various population topologies. One of common population topologies suggested was a ring topology, where each particle on the population array is only allowed to interact with its two immediate neighbours. Among all topologies studied, the ring topology was considered to be "*the slowest, most indirect communication pattern*", whereas the *gbest* PSO represents "*the most immediate communication possible*".

Clearly the ring topology is desirable for locating multiple optima, because ideally we would like to have individuals to search thoroughly in its local neighbourhood before propagating the information throughout the population. The consequence of any quicker than necessary propagation would result in the population converging onto a single optimum (like *gbest* PSO).

As we will demonstrate in the following sections, the ring topology is able to provide the right amount of communication needed for inducing stable niching behaviour.

4.1 Memory-Swarm vs. Explorer-Swarm

In PSO, interactions among particles play an important role in particles' behaviour. A distinct feature of PSO (which is different from many EAs) is that each particle carries a *memory* of its own, i.e., its personal best. We can never underestimate the significance of using *local memory.* As remarked by Clerc in [11], a swarm can be viewed as comprising of two sub-swarms according to their differences in functionality. The first group, *explorer-swarm*, is composed of particles moving around in large step sizes and more frequently, each strongly influenced by its velocity and its previous position (see equation (1) and (2)). The *explorer-swarm* is more effective in exploring more broadly the search space. The second group, *memory-swarm*, consists of personal bests of all particles. This *memory-swarm* is more stable than the *explorer-swarm* because personal bests

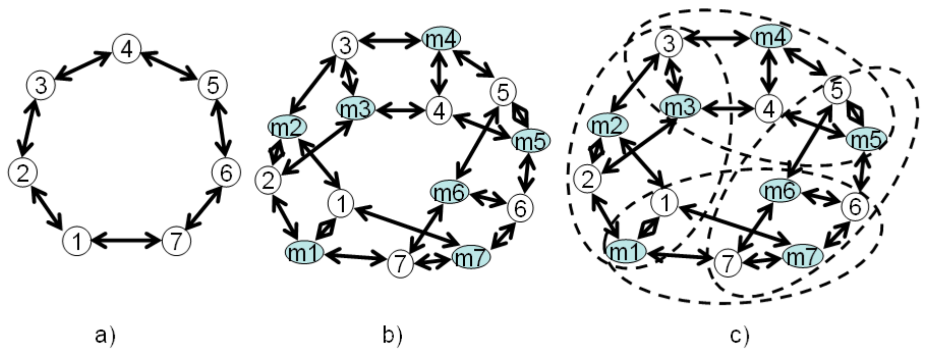


Fig. 4. a) The ring topology used in a conventional EA. Each member interacts only with its immediate left and right neighbours, with no *local memory* used; b) Graph of influence for a *lbest* PSO using the same ring topology (see also p.89 in [11]). Each particle possesses a *local memory*; c) The same as b) but also showing the overlapping subpopulations, each consisting of a particle and its two immediate neighbours, and their corresponding memories.

represent positions of only the best positions found so far by individual particles. The *memory-swarm* is more effective in retaining better positions found so far by the swarm as a whole.

Fig. 4 a) shows an example of a conventional EA using a ring topology with a population of 7 individuals. Fig. 4 b) shows a swarm of 7 particles using a ring topology, as illustrated by using a 'graph of influence' as suggested by Clerc [11]. The 'graph of influence' can be used to explicitly demonstrate the source and receiver of influence for each particle in a swarm. A particle that informs another particle is called 'informant'. Here the *explorer-swarm* consists of particles as marked from numbers 1 to 7, and the *memory-swarm* consists of particles as marked from *m1* to *m7*. Each particle has 3 informants, from two neighbouring particles' memories and its own memory. Each particle's memory also has 3 informants, from two neighbouring particles and the particle itself. In stark contrast, Fig. 4 a) shows that no local memories are used in a conventional EA using a ring topology.

The idea of memory-swarm and explorer-swarm inspired us to develop effective PSO niching algorithms. With an aim to locate and maintain multiple optima, the more stable personal best positions retained in the *memory-swarm* can be used as the 'anchor' points, providing the best positions found so far. Meanwhile, each of these positions can be further improved by the more exploratory particles in the *explorer-swarm*.

4.2 *lbest* PSO Using a Ring Topology

As shown in Fig. 4, in a *lbest* PSO with a ring topology, each particle interacts only with its immediate neighbours. An implementation of such a *lbest* PSO

```
Randomly generate an initial population
repeat
    for $i = 1$ to Population Size do
        if $fit(\mathbf{x}_i) > fit(\mathbf{p}_i)$ then $\mathbf{p}_i = \mathbf{x}_i$;
    end
    for $i = 1$ to Population Size do
        $\mathbf{p}_{n,i}$ = neighbourhoodBest($\mathbf{p}_{i-1}, \mathbf{p}_i, \mathbf{p}_{i+1}$);
    end
    for $i = 1$ to Population Size do
        Equation (1);
        Equation (2);
    end
until termination criterion is met ;
```

Algorithm 4. The pseudocode of a *lbest* PSO using a ring topology. Note that in equation (1), $\mathbf{p}_g$ should be replaced by the i-th particle's neighbourhood best $\mathbf{p}_{n,i}$.

using a ring topology is provided in Algorithm 4. Note that we can conveniently use population indices to identify the left and right neighbours of each particle. Here we assume a 'wrap-around' ring topology, i.e., the first particle is the neighbour of the last particle and vice versa. The *neighbourhoodBest*() function returns the best-fit personal best in the i-th neighbourhood, which is recorded as $\mathbf{p}_{n,i}$ (denoting the neighbourhood best for the i-th particle). This $\mathbf{p}_{n,i}$ is then used as the *local leader* when updating the i-th particle in Equation (1) and (2).

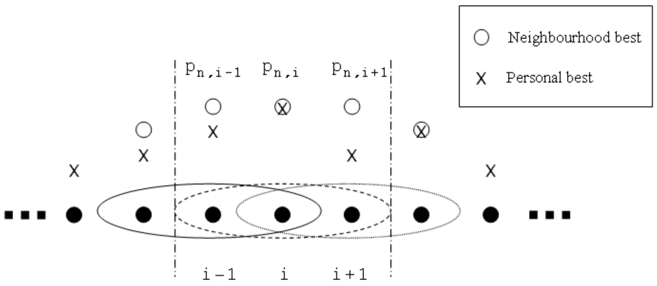


Fig. 5. A ring topology with each member interacting with its 2 immediate neighbours (left and right). Local neighbourhoods are overlapped with each other. The i-th particle's neighbourhood best $\mathbf{p}_{n,i}$ is the same as those of its 2 immediate neighbouring particles, but differs from those particles in the neighbourhoods further out.

Note that different particles residing on the ring can have different $\mathbf{p}_n$[1], and they do not necessarily converge into a single value over time. As illustrated in

[1] We use $\mathbf{p}_n$ to denote a non-specific 'neighbourhood best'.

Fig. 5, the ring topology not only provides a mechanism to slow down information propagation in the particle population, but also allows different neighbourhood bests to *coexist* (rather than becoming homogeneous) over time. This is because a particle's $\mathbf{p}_n$ can only be updated if there is a better personal best in its neighbourhood, but not by a better $\mathbf{p}_n$ of its neighbouring particle. Assuming that particles from the initial population are uniformly distributed across the search space, niches can naturally emerge as a result of the coexistence of multiple $\mathbf{p}_n$ positions being the local attraction points for the particles in the population. With a reasonable population size, such a *lbest* PSO is able to form stable niches around the identified neighbourhood bests $\mathbf{p}_n$.

Apart from its simplicity, the ring topology *lbest* PSO does not require any prior knowledge of (neither the need to specify) any niching parameters, e.g., the niche radius or the number of peaks, since niches emerge naturally from the initial population. The complexity of the algorithm is only $\mathcal{O}(N)$ (where N is the population size), as the calculation to obtain a neighbourhood best is only done locally from each particle's local neighbourhood.

4.3 Numerical Examples

To evaluate the niching ability of the above *lbest* PSO with a ring topology, we used 3 multimodal optimization test functions of different characteristics.[2] f_1 *Equal Maxima* has 5 evenly spaced global maxima, whereas f_2 *Uneven Maxima* has 5 global maxima unevenly spaced. f_3 *Inverted Shubert function* is the inverted Shubert function, as shown in Fig. 1, the inverted Shubert 2D function has 9 groups of global optima, with 2 very close global optima in each group. For n-dimensional Shubert function, there are $n \cdot 3^n$ global optima unevenly distributed. These global optima are divided into 3^n groups, with each group having n global optima being close to each other. For f_3 Shubert 3D, there are 81 global optima in 27 groups; whereas for f_3 Shubert 4D, there are 324 global optima in 81 groups. f_3 will pose a serious challenge to any niching algorithm relying on a fixed niche radius parameter.

The *lbest* PSO with a ring topology as described above has overlapping local neighbourhoods. To further restrain the influence from a few dominant $\mathbf{p}_n$ points, we could reduce the neighbourhood size or even completely remove the overlaps. In our experiments, we used the following ring topology based PSO variants:

- **r3pso**: a *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right (as in Fig. 5);
- **r2pso**: a *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right;
- **r3pso-lhc**: the same as **r3pso**, but with no overlapping neigbourhoods. Basically multiple PSOs search in parallel, like local hill climbers.
- **r2pso-lhc**: the same as **r3pso-lhr**, but with each member interacts with only its next member on the population array.

[2] These 3 functions are also described in [23].

Table 1. Success rates.

fnc	**r2pso**	**r3pso**	**r2pso-lhc**	**r3pso-lhc**	**SPSO**
f_1	98%	**100%**	**100%**	**100%**	**100%**
f_2	**100%**	**100%**	**100%**	**100%**	**100%**
f_3(2D)	94%	**100%**	**100%**	98%	60%

For any particle with its $\mathbf{x}_i$ exceeding the boundary of the variable range, its position is reset to a value which is twice of the right (or left boundary) subtracting $\mathbf{x}_i$.

The performance of the above PSO variants were compared with SPSO [26], which is a typical niching algorithm requiring a user to pre-specify a niche radius parameter.

To compare the performance of niching algorithms, we first allow a user to specify a *level of accuracy* (typically $0 < \epsilon \leq 1$), i.e., how close the computed solutions to the expected optimal solutions are. If the distance from a computed solution to an expected optimum is below the specified ϵ, then we can consider the optimum is found. For all comparing niching algorithms in this paper, we used SPSO's procedure for identifying species seeds (as described in the previous section) to check if a niching algorithm has located all expected global optima. Note that this procedure was only used for the purpose of performance measurement, but not for optimization in the proposed PSO niching methods, with the only exception of SPSO itself.

All PSO niching algorithms' performance were measured in terms of *success rate*, i.e., the percentage of runs in which all global optima are successfully located, for a given number of evaluations in each run.

4.4 Results and Discussion

For the relatively simple f_1 and f_2, a population size of 50 was used. The PSO niching variants were run until all the known global optima were found, or a maximum of 100,000 evaluations was reached. For the more challenging f_3 2D and 3D, a population size of 500 was used. For f_3 3D, we allowed a maximum of 200,000 evaluations for each run. For f_3 4D, a population size of 1000 was used, and we allowed a maximum of 400,000 evaluations. All results were averaged over 50 runs. For all PSO niching methods (except SPSO) ϵ and r (niche radius) were used purely for the purpose of performance measurement. In order to measure more accurately each niching algorithm's ability in forming niches in the vicinities of all known global optima, for f_1 and f_2, both ϵ and r were set to 0.01. For f_3 2D, ϵ was set to 0.1, and for f_3 3D and 4D, ϵ was set to 0.2. For all f_3 2D, 3D and 4D, r was set to 0.5.

Table 1 presents the success rates on f_1, f_2 and f_3 2D. On f_1 nd f_2, almost all comparing PSOs achieved a 100% success rate. However, for the more challenging f_3 2D, SPSO did not perform very well, whereas the ring topology PSOs achieved success rates greater than 90%. Bear in mind that SPSO was tuned with the

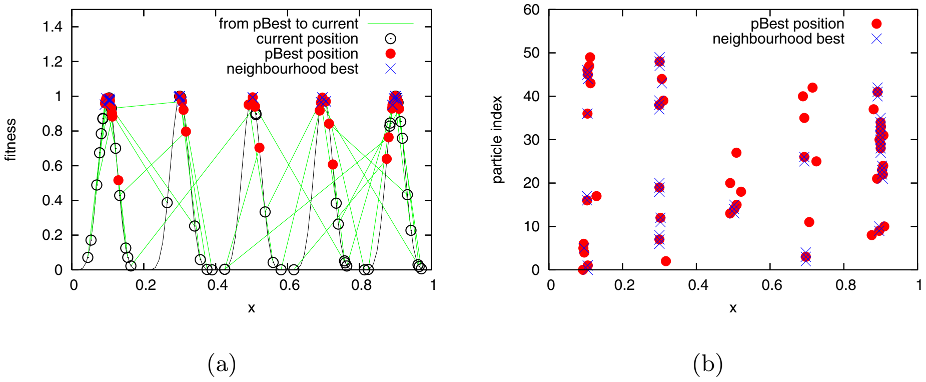


Fig. 6. a) Niches formed when using **r3pso** variant on the f_1 at iteration 15 (a population size of 50 was used); b) Particles' $\mathbf{p}_i$ and their $\mathbf{p}_n$ on the population array at iteration 15, corresponding to the run in a).

Table 2. Averaged peak ratios on f_{11} Inverted Shubert 3D and 4d over 50 runs.

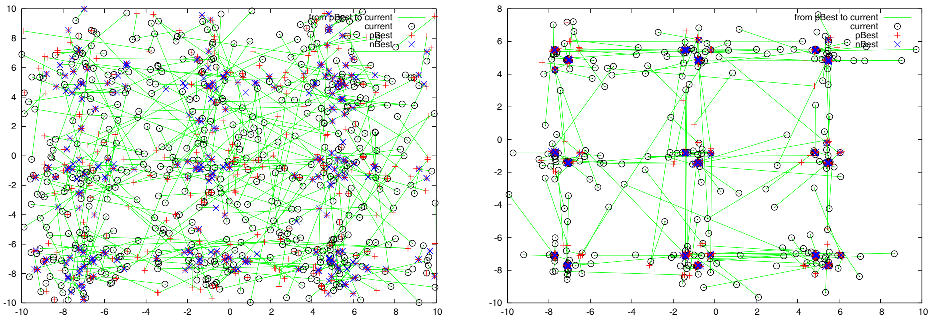
fnc	ϵ	r	r2pso	r3pso	r2pso-lhc	r3pso-lhc	SPSO
f_3 (3D)	0.2	0.5	0.16	0.61	0.27	**0.66**	0.01
f_3 (4D)	0.2	0.5	0.00	**0.25**	0.00	0.14	0.00

optimal niche radius, whereas the ring topology based PSOs did not depend on any niching parameters, showing greater robustness.

Fig. 6 a) shows an example of running **r3pso** on f_1. At iteration 15, all 5 global peaks were located by the $\mathbf{p}_n$ points identified for individual particles on the population array. Although particles' $\mathbf{x}_i$ points (i.e., current positions) tended to be more exploratory oscillating around peaks, their $\mathbf{p}_n$ points converged stably on the tips of the peaks, even if we ran the model for a large number of iterations. Niches formed from neighbouring particles (as shown by their indices on the population array) are clearly visible in Fig. 6 b). It can be also observed that for each of the 5 peaks, **r3pso** formed multiple small niches centered around the $\mathbf{p}_n$ points.

Fig. 7 shows that **r3pso** was able to locate all 18 global peaks on f_3 the inverted Shubert 2D by iteration 75 in a single run. Multiple emerged niches are clearly visible.

For the more challenging f_3 Inverted Shubert 3D and 4D functions, since no run can find all peaks, hence we used *averaged peak ratio* (instead of *success rate*) as the performance measure. Peak ratio measures the percentage of global peaks found in a single run. Table 2 shows the averaged peak ratios on f_3 Inverted



(a) iteration 10 (b) iteration 75

Fig. 7. The niching behaviour of the **r3pso** (with a population size of 500) on f_3 the inverted Shubert 2D function at iteration 10 and 75 of a run.

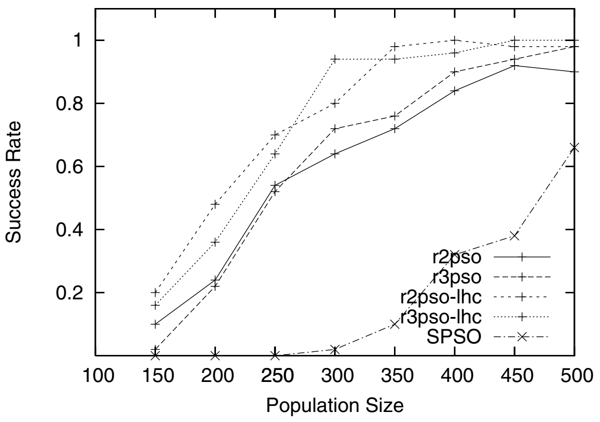


Fig. 8. Success rates for varying population sizes on f_3 the inverted Shubert 2D function.

Shubert 3D and 4D over 50 runs. As can be seen in Table 2, **r3pso** and **r3pso-lhc** are the best performers, whereas SPSO is the worst. **r2pso** and **r2pso-lhc** were able to find a few global peaks on f_3 3D, but failed to find almost any peaks on f_3 4D.

4.5 Effect of Varying Population Size

For the proposed ring topology based PSOs, the only parameter that needs to be specified is population size. Given a reasonably large population size, these PSOs are able to locate all global optima reliably. Fig. 8 shows that on f_3 2D, with a population size of 450 or above, the ring topology based PSOs achieved 90% or above success rates. In contrast, even with a population size of 500, SPSO only managed to achieve 60% success rate. Another similar niching algorithm, SCGA [20], which also required a user to specify a niche radius parameter, needed a population size of 1000 or above in order to locate all 18 global optima.

It is worth noting that the local hill-climber variants **r2pso-lhc** and **r3pso-lhc** performed better than **r2pso** and **r3pso** on f_3 2D. This indicates that when handling problems with a large number of global optima, it might be more effective to have multiple local hill climbers each optimizing independently than a niching algorithm working with a single large population.

5 Conclusions

Niching methods have been developed mostly in the context of EAs, and have been around for more than two decades. Recent advances in Swarm Intelligence and in particular PSO has made possible to design novel and competent niching methods for multimodal optimization. This chapter has presented a survey of several state-of-the-art PSO niching algorithms, and described how some of the challenging issues faced by classic niching methods can be addressed. Apart from the fact that existing niching methods developed in the early days of EAs can be easily incorporated into a PSO, more importantly, it has been shown here that the inherent characteristics of PSO can be utilized to design highly competitive niching algorithms. In particular, it is shown that in a *lbest* PSO, *local memory* and *slow communication topology* are the two key elements for its success as an effective niching algorithms. In fact it is foreseeable that other population based stochastic optimization methods characterized by these two key elements can be also used to induce stable niching behaviour.

In future, we will be interested in investigating how to increase the search capability of small niches so that the performance of these niches will scale well with increasing dimensions, since *lbest* PSO niching algorithms tend to generate multiple small niches. Ideally a function generator suitable for this kind of evaluation will need to offer controllable features such as the number global optima and local optima, which are independent from the number of dimensions. One recently proposed function generator for multimodal function optimization in [32] seems to be a promising tool for this purpose. We will be also interested in developing techniques to adapt or self-adapt the population size, as this is the only parameter that still needs to be supplied by a user. Anther interesting research topic will be to apply the *lbest* niching PSO to tracking multiple peaks in a dynamic environment [26].

References

1. Beasley, D., Bull, D.R., Martin, R.R.: A sequential niche technique for multimodal function optimization. Evolutionary Computation 1(2), 101–125 (1993), `citeseer.ist.psu.edu/beasley93sequential.html`

2. Bessaou, M., Pétrowski, A., Siarry, P.: Island model cooperating with speciation for multimodal optimization. In: Deb, K., Rudolph, G., Lutton, E., Merelo, J.J., Schoenauer, M., Schwefel, H.-P., Yao, X. (eds.) PPSN 2000. LNCS, vol. 1917, pp. 16–20. Springer, Heidelberg (2000), `citeseer.ist.psu.edu/bessaou00island.html`
3. Bird, S.: Adaptive techniques for enhancing the robustness and performance of speciated psos in multimodal environments, phd thesis. Ph.D. dissertation, RMIT University, Melbourne, Australia (2008)
4. Bird, S., Li, X.: Adaptively choosing niching parameters in a PSO. In: Cattolico, M. (ed.) Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2006, Seattle, Washington, USA, July 8-12, pp. 3–10. ACM, New York (2006), `http://doi.acm.org/10.1145/1143997.1143999`
5. Bird, S., Li, X.: Enhancing the robustness of a speciation-based PSO. In: Yen, G.G. (ed.) Proceedings of the 2006 IEEE Congress on Evolutionary Computation, July 16-21, pp. 843–850. IEEE Press, Vancouver (2006), `http://ieeexplore.ieee.org/servlet/opac?punumber=11108`
6. Bird, S., Li, X.: Using regression to improve local convergence. In: Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, pp. 1555–1562 (2007)
7. Blackwell, T., Branke, J., Li, X.: Particle swarms for dynamic optimization problems. In: Blum, C., Merkle, D. (eds.) Swarm Intelligence - Introduction and Applications, pp. 193–217. Springer, Heidelberg (2008)
8. Blackwell, T.M., Branke, J.: Multi-swarm optimization in dynamic environments. In: Raidl, G.R., Cagnoni, S., Branke, J., Corne, D.W., Drechsler, R., Jin, Y., Johnson, C.G., Machado, P., Marchiori, E., Rothlauf, F., Smith, G.D., Squillero, G. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 489–500. Springer, Heidelberg (2004)
9. Brits, R., Negelbrecht, A., van den Bergh, F.: Solving systems of unconstrained equations using particle swarm optimizers. In: Proc. of the IEEE Conference on Systems, Man, Cybernetics, October 2002, pp. 102–107 (2002)
10. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. on Evol. Comput. 6, 58–73 (February 2002)
11. Clerc, M.: Particle Swarm Optimization. ISTE Ltd., London (2006)
12. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, University of Michigan (1975)
13. Goldberg, D.E., Deb, K., Horn, J.: Massive multimodality, deception, and genetic algorithms. In: Männer, R., Manderick, B. (eds.) PPSN 2. Elsevier Science Publishers, B. V., Amsterdam (1992), `citeseer.ist.psu.edu/goldberg92massive.html`
14. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Grefenstette, J. (ed.) Proc. of the Second International Conference on Genetic Algorithms, pp. 41–49 (1987)
15. Harik, G.R.: Finding multimodal solutions using restricted tournament selection. In: Eshelman, L. (ed.) Proc. of the Sixth International Conference on Genetic Algorithms, pp. 24–31. Morgan Kaufmann, San Francisco (1995), `citeseer.ist.psu.edu/harik95finding.html`
16. Holland, J.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975)

17. Kennedy, J.: Stereotyping: Improving particle swarm performance with cluster analysis. In: Proc. of IEEE Int. Conf. Evolutionary Computation, pp. 303–308 (2000)
18. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann, San Francisco (2001)
19. Kennedy, J., Mendes, R.: Population structure and particle swarm performance. In: Proc. of the 2002 Congress on Evolutionary Computation, pp. 1671–1675 (2002)
20. Li, J.-P., Balazs, M.E., Parks, G.T., Clarkson, P.J.: A species conserving genetic algorithm for multimodal function optimization. Evol. Comput. 10(3), 207–234 (2002)
21. Li, X.: Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 105–116. Springer, Heidelberg (2004)
22. Li, X.: Multimodal function optimization based on fitness-euclidean distance ratio. In: Thierens, D. (ed.) Proc. of Genetic and Evolutionary Computation Conference 2007, pp. 78–85 (2007)
23. Li, X.: Niching without niching parameters: Particle swarm optimization using a ring topology. IEEE Trans. on Evol. Comput. 14(1), 150–169 (2010)
24. Mahfoud, S.W.: Crowding and preselection revisited. In: Männer, R., Manderick, B. (eds.) Parallel Problem Solving From Nature 2, pp. 27–36. North-Holland, Amsterdam (1992), `citeseer.ist.psu.edu/mahfoud92crowding.html`
25. Mahfoud, S.W.: Niching methods for genetic algorithms. Ph.D. dissertation, Urbana, IL, USA (1995), `http://citeseer.ist.psu.edu/mahfoud95niching.html`
26. Parrott, D., Li, X.: Locating and tracking multiple dynamic optima by a particle swarm model using speciation. IEEE Trans. on Evol. Comput. 10(4), 440–458 (2006)
27. Parsopoulos, K., Vrahatis, M.: Modification of the particle swarm optimizer for locating all the global minima. In: Kurkova, R.N.M.K.V., Steele, N. (eds.) Artificial Neural Networks and Genetic Algorithms, pp. 324–327. Springer, Heidelberg (2001)
28. Parsopoulos, K., Vrahatis, M.: On the computation of all global minimizers through particle swarm optimization. IEEE Trans. on Evol. Compu. 8(3), 211–224 (2004)
29. Passaro, A., Starita, A.: Particle swarm optimization for multimodal functions: a clustering approach. J. Artif. Evol. App. 2008, 1–15 (2008)
30. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proc. of the 3rd IEEE International Conference on Evolutionary Computation, pp. 798–803 (1996)
31. Brits, A.E.R., van den Bergh, F.: A niching particle swarm optimizer. In: Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002 (SEAL 2002), pp. 692–696 (2002)
32. Rönkkönen, J., Li, X., Kyrki, V., Lampinen, J.: A generator for multimodal test functions with multiple global optima. In: Li, X., Kirley, M., Zhang, M., Green, D., Ciesielski, V., Abbass, H.A., Michalewicz, Z., Hendtlass, T., Deb, K., Tan, K.C., Branke, J., Shi, Y. (eds.) SEAL 2008. LNCS, vol. 5361, pp. 239–248. Springer, Heidelberg (2008)
33. Schoeman, I.: Niching in particle swarm optimization, phd thesis. Ph.D. dissertation, University of Pretoria, Pretoria, South Africa (2009)

34. Schoeman, I., Engelbrecht, A.: Using vector operations to identify niches for particle swarm optimization. In: Proc. of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, Singapore, pp. 361–366 (2004)
35. Schoeman, I., Engelbrecht, A.: A parallel vector-based particle swarm optimizer. In: Proc. of the 7th International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA 2005), Coimbra, Portugal (2005)
36. Schwarz, G.: Estimating the dimension of a model. Annals of Statistics 6(2), 461–464 (1978)
37. Yin, X., Germay, N.: A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization. In: The International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 450–457 (1993)