Niching Without Niching Parameters: Particle Swarm Optimization Using a Ring Topology

Xiaodong Li, *Senior Member, IEEE*

***Abstract*—Niching is an important technique for multimodal optimization. Most existing niching methods require specification of certain niching parameters in order to perform well. These niching parameters, often used to inform a niching algorithm how far apart between two closest optima or the number of optima in the search space, are typically difficult to set as they are problem dependent. This paper describes a simple yet effective niching algorithm, a particle swarm optimization (PSO) algorithm using a ring neighborhood topology, which does not require any niching parameters. A PSO algorithm using the ring topology 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, PSO algorithms using the ring topology are able to form stable niches across different local neighborhoods, eventually locating multiple global/local optima. The complexity of these niching algorithms is only $\mathcal{O}(N)$, where N is the population size. Experimental results suggest that PSO algorithms using the ring topology are able to provide superior and more consistent performance over some existing PSO niching algorithms that require niching parameters.**

***Index Terms*—Evolutionary computation, multimodal optimization, niching algorithms, particle swarm optimization (PSO), swarm intelligence.**

I. 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 [1]. 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 an optimization problem with multiple global and local optima, 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 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 may be reduced.

The success of EAs in real-world applications has also been accompanied by their uses of niching methods. For example, classification problems in machine learning can be mapped to multimodal optimization problems, and hence be treated by an EA employing a niching method [2]. A niching GA was also applied to the problem of the inversion of teleseismic waves [3]. In evolutionary multiobjective optimization, niching methods are often used to maintain solution diversity [4].

Many niching methods have been proposed in the EA literature. Some representative examples include crowding [5], deterministic crowding [6], fitness sharing [7], derating [8], restricted tournament selection [9], parallelization [10], clustering [11], clearing [12], and speciation [13]. Niching methods have also been incorporated into PSO variants to enhance their ability to handle multimodal optimization problems including NichePSO [14] and SPSO [15], [16]. Most of these niching methods, however, have difficulties that need to be overcome before they can be applied successfully to real-world multimodal problems. Some identified issues include the following:

1) 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 [7]. Other uses of niching parameters include *crowding factor* in *crowding method* [5], the window size w in *restricted tournament selection* [9], or the number of clusters in *k-means clustering methods* [11], [17]. Further discussion on the issue of niching parameters is provided in Section III.
2) 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 [6]. A good niching algorithm should be able to form and maintain stable subpopulations over the run.

Manuscript received May 13, 2008; revised March 2, 2009. Current version published January 29, 2010.

The author is with the School of Computer Science and Information Technology, Royal Melbourne Institute of Technology, Melbourne 3001, Australia (e-mail: xiaodong.li@rmit.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2026270

3) 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 [2]. How can we minimize such detrimental crossover operations across different niches?
4) Some existing niching methods are designed only for locating all global optima, while ignoring local optima. Examples include the sequential niche GA (SNGA) [8], clearing [12], SCGA [13], NichePSO [14], and SPSO [15], [16]. However, it might be desirable to obtain both global and local optima in a single run.
5) 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, probably only with the exception of deterministic crowding (DC) [6]. However, several studies found that DC required a higher number of evaluations [18] and was unable to maintain a proportional distribution of individuals across niches [19].

To tackle the above-mentioned issues, this research aims to develop niching algorithms that:

1) do not require specification of any niching parameters such as how far apart between two optima, or the number of optima;
2) are able to locate multiple optima and maintain these found optima until the end of a run;
3) are able to locate multiple global optima, as well as local optima, as required by a user;
4) have low computational complexity.

Two important criteria by which we can measure the success of a niching method are whether a niching algorithm can find all desired optima including global and/or local optima, and whether it can maintain multiple subpopulations stably over a run.

This research to develop niching algorithms without niching parameters is strongly motivated by the recent development of particle swarm optimization (PSO) [1], [20], [21]. PSO has some attractive characteristics that make it an ideal optimization algorithm to induce niching behaviors. In a canonical PSO, each particle in the population knows its current position in the search space, as well as the best position it has visited so far (so called personal best position). As described in more detail later on, the notion of *memory* and the viewpoint of a swarm constituting an *explorer-swarm* and a *memory-swarm* play an important role in the niching PSO algorithms described in this paper. The basic idea is the following—the personal bests of all particles in the population can be used to form a *memory-swarm* providing a stable network retaining the best positions 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 toward a fitter local best only within its vicinity of the search space. As search continues, multiple niches (or subpopulations) are formed around optima in parallel. Eventually multiple optima are found. Since niches are formed naturally, there is no need to specify any niching parameter in these kinds of niching methods.

This paper describes an attempt to eliminate the need of specifying any niching parameters. We demonstrate that by mapping a PSOs population onto a ring topology, its initial population can be naturally divided into multiple subpopulations, each operating as a separate PSO with its own local neighborhood best. Since these subpopulations are only loosely connected, the speed of convergence is greatly reduced. Most importantly, we demonstrate that the ring topology based PSO can be used as an effective niching method for maintaining stable subpopulations (or niches); therefore, it can reliably locate multiple global and/or local optima.

Using a ring topology for PSO algorithms is not new. In fact, it was described in one of the first papers on PSO by Eberhart and Kennedy [22]. Several more recent studies include [21], [23]. However, the ring topology based PSO algorithms were primarily used in these studies with an aim to locate a single global optimum, rather than multiple global optima (which is the optimization goal of a niching method). It was found in several recent studies that *lbest* PSOs (i.e., PSOs employing some local neighborhood topology) were unable to induce stable niching behaviors or were inefficient in doing so [18], [24]. This paper differs from these previous studies by showing that ring topology based PSO algorithms are capable of inducing stable niching behaviors and they are robust and effective niching methods. In particular, one key advantage of such PSO niching algorithms is that there is no need to specify any niching parameters that are often required in traditional niching methods.

The paper is organized as follows. Section II provides a review of the classic niching methods, followed by Section III discussing the issue of niching parameters. Section IV then gives an introduction to PSO and existing PSO niching methods. Furthermore, we describe the notion of *memory-swarm* and *explorer-swarm*, which have direct relevance to the ring topology based PSO niching algorithms described in this paper. Section V discusses first why the ring topology is appropriate and then describes in detail the ring topology based PSO niching algorithms. Convergence behaviors of these niching algorithms are also studied in this section. Section VI describes the experimental setup and performance measures, followed by detailed numerical results in Section VII. Finally, Section VIII gives the concluding remarks.

II. Revisiting Classic Niching Methods

Just like EAs 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. As Mahoud described [2], *"A niching method must be able to form and maintain multiple, diverse, final solutions, whether these solutions are of identical fitness or of varying fitness. A niching method must be able to maintain these solutions for an exponential to infinite time period, with respect to population size."* Cavicchio's dissertation [25] was probably one of the first studies attempting to induce niching behavior in a GA. Cavicchio proposed several schemes in which offspring directly replaced the parents that produce them. These so-called *preselection* schemes were later generalized by De Jong in a scheme called *crowding*. De Jong's *crowding* was initially designed only to preserve population diversity. 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 *crowding factor* (CF) is commonly used to determine the size of the sample. In [6], Mahfoud carefully examined both *crowding* and *preselection* and found that De Jong's crowding method was unable to maintain more than two peaks of a five peaks fitness landscape due to stochastic replacement errors. Mahfoud then made several modifications to *crowding* to reduce replacement errors, restore selection pressure, and also eliminate the crowding factor parameter. The resulting algorithm, *deterministic crowding*, was able to locate and maintain multiple peaks.

Another approach to induce niching behavior in an EA is *fitness sharing*, which is probably the most widely used niching method. The sharing concept was originally introduced by Holland [26] and then adopted by Goldberg and Richardson as a mechanism to divide the population into different subpopulations [7] 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 in an EA to degrade an individual's fitness based on the presence of other neighboring individuals. During selection, many individuals in the same neighborhood would degrade each other's fitness, thereby limiting the number of individuals occupying the same niche. For example, if there are two individuals, i and j, and the distance between the two is d_{ij}, then a sharing function $sh(d_{ij}) = 1 - (\frac{d_{ij}}{\sigma_{\text{share}}})^{\alpha}$ can be used to compute the shared value for both i and j, if $d_{ij} \leq \sigma_{\text{share}}$ (where σ_{share} is the estimated niche radius), and this value is added to their niche counts m_i and m_j respectively. If $d_{ij} > \sigma_{\text{share}}$, then i and j do not degrade each other's fitness, i.e., $sh(d_{ij}) = 0$. For an individual i, the niche count is calculated as $m_i = \sum_{j=1}^{N} sh(d_{ij})$, i.e., the sum of its shared values computed over the population of size N. Finally the shared fitness value for individual i is $\frac{f_i}{m_i}$. 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 niche radius σ_{share} and the scaling factor α [27], [28], without prior knowledge of the problems. For many real-world problems, it is not uncommon that such domain specific knowledge is often unavailable.

Apart from the above, many other niching methods have also been developed over the years, including *derating* [8], *deterministic crowding* [6], *restricted tournament selection* [9], *parallelization* [10], *clustering* [11], *clearing* [12] and *speciation* [13]. In a recent work, Singh and Deb [29] proposed a *modified clearing* method where individuals not qualified as the best are redistributed to 1.5 times σ_{clear} or more distance away. The algorithm showed good performance compared with several classic niching EAs. However, the performance of *modified clearing* still depends on how well σ_{clear} is set. Niching methods have also been developed for PSOs, such as NichePSO [14] and SPSO [16]. More discussion on existing PSO niching algorithms will be given in Section IV-A.

III. 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* [7], the species distance σ_s in *species conserving GA* (SCGA) [13], the distance measure σ_{clear} in *clearing*, [12] and the species radius r_s in the *speciation-based PSO* (SPSO) [16]. Although different terminologies are used, what is in common is that they all describe some kind of distance value by which a niche can be defined. Therefore, in this paper, these terms will be commonly referred to as *niche radius*.

Fig. 1 shows two examples of challenging fitness landscapes that would be difficult for any niching algorithms using a uniform niche radius value. The inverted Shubert 2-D function has nine pairs of global peaks and numerous local peaks. Within each pair, two global peaks are very close to each other but peaks from different pairs are further away. The inverted Vincent 2-D function has 36 global peaks with distances between these peaks varying greatly. A niching algorithm relying on a fixed niche radius value to determine an individual'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. Dependence on a fixed niche radius is a major drawback for niching methods that rely on such a parameter. For example, on the inverted Shubert 2-D function, 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 [13]. For Shubert 3-D, SCGA used a population size of 4000 in order to locate all 81 global peaks. As the 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 [16] suffers from the same problem.

In an early work, Jelasity and Dombi [30] attempted to tackle the *niche radius* problem where they observed that a

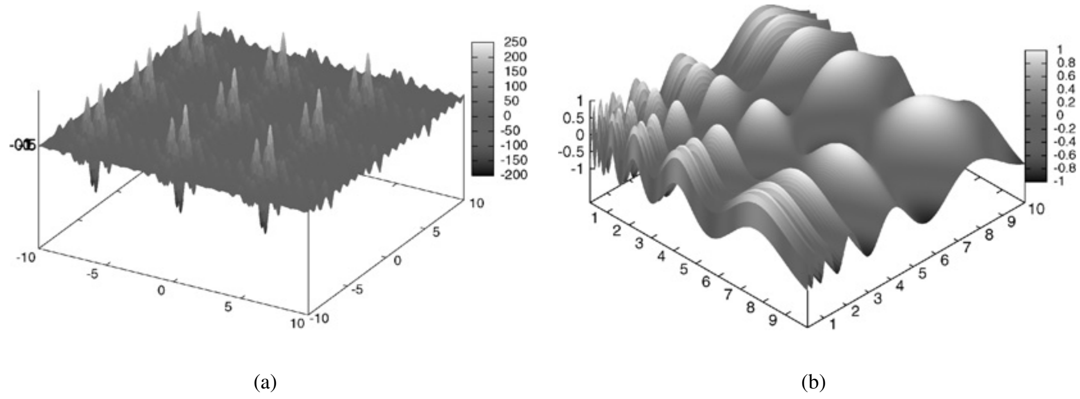


Fig. 1. Challenging functions for niching methods using a uniform niche radius value. (a) Inverted Shubert 2-D function. (b) Inverted Vincent 2-D function.

niching method such as fitness sharing could not distinguish optima that are much closer to each other than the provided niche radius value. Instead of using a fixed niche radius, Jelasity and Dombi proposed to use a radius function and a cooling procedure similar to simulated annealing. However, their method called GAS (S for species) introduced several new parameters that have to be specified by a user, such as the radius function *R*, the number of evaluations, the maximal number of species, and the number of steps for cooling. It seems that for the user to specify these parameters is also rather challenging.

Shir and Bäck [31] also attempted to address the problem of using a fixed niche radius. Inspired by the notion of self-adaptation in evolutionary strategy, in Shir and Bäck's CMA-ES niching algorithm, each individual is allowed to adapt its own niche radius value along with other adaptive strategy parameters. The adaptation of individual niche radii provided better performances than those niching methods using a fixed niche radius. The downside of their method is again the introduction of some new parameters, which may affect the performance of the algorithm, e.g., the number of expected optima and learning coefficients.

Recent development of PSO niching methods by Bird and Li [32], [33] aimed to reduce the sensitivity of the SPSO to the niche radius values. However, either this parameter still remains (though made more robust), or several new parameters are introduced.

Sometimes niching parameters can be under different disguises, such as the *crowding factor* in *crowding method* [5], the window size w in *restricted tournament selection* [9], or the number of clusters in *k-means clustering methods* [11], [17]. The performance of these EAs depends very much on how these parameters are specified. Unfortunately, in many real-world problems such prior knowledge is often unavailable. It would be desirable if a user can be completely freed from specifying any niching parameters.

In this paper, we will demonstrate that a PSO using a ring topology is able to induce stable niching behavior without using any niching parameters. Furthermore, the proposed niching method does not introduce new parameters. Therefore, it offers distinct advantages over the above-mentioned niching methods.

IV. PSO

PSO is a swarm intelligence technique originally developed from studies of social behaviors of animals or insects, e.g., bird flocking or fish schooling [1]. Since its inception in 1995 [1], PSO has gained increasing popularity among researchers and practitioners as a robust and efficient technique for solving complex and difficult optimization problems.

Like an EA, PSO is population-based. However, PSO differs from EAs in the way it manipulates each particle (i.e., a candidate solution) in the population. Instead of using evolutionary operators such as crossover and mutation, PSO modifies each particle's position in the search space, based on its velocity, some previous best positions it has found so far, and previous best positions found by its neighbors.

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 us use $\vec{\mathbf{v}}_i$ to denote the velocity of the ith particle in the swarm, $\vec{\mathbf{x}}_i$ its position, $\vec{\mathbf{p}}_i$ the best position it has found so far, and $\vec{\mathbf{p}}_g$ the best position found from the entire swarm (so-called global best). $\vec{\mathbf{v}}_i$ and $\vec{\mathbf{x}}_i$ of the ith particle in the swarm are updated according to the following two equations [34]:

$$\begin{aligned} \vec{\mathbf{v}}_i &\leftarrow \chi(\vec{\mathbf{v}}_i + \vec{\mathbf{R}}_1[0,\varphi_1] \otimes (\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) + \\ &\quad \vec{\mathbf{R}}_2[0,\varphi_2] \otimes (\vec{\mathbf{p}}_g - \vec{\mathbf{x}}_i)) \qquad (1) \\ \vec{\mathbf{x}}_i &\leftarrow \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i \qquad (2) \end{aligned}$$

where $\vec{\mathbf{R}}_1[0,\varphi_1]$ and $\vec{\mathbf{R}}_2[0,\varphi_2]$ are two separate functions each returning a vector comprising random values uniformly generated in the ranges $[0,\varphi_1]$ 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 [34] 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 $\varphi = \varphi_1 + \varphi_2 = 4.1$.

Two common approaches of choosing $\vec{\mathbf{p}}_g$ in (1) are known as *gbest* and *lbest* methods. In a *gbest* PSO, the position of each particle is influenced by the best-fit particle from 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 studied PSOs with various population topologies [23], and have shown that certain population topologies could give superior performance over certain optimization functions.

A. PSO Niching Methods

Several niching methods have been developed within the framework of a PSO. Parsopoulos and Vrahitis [35] 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 [36], similar to the derating method used in SNGA [8]. 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, their PSO was able to locate all the global optima of some selected test functions successfully. However, this *stretching* method introduces several new issues, including the difficulty in specifying several new parameters used in the *stretching* function, and the risk of introducing false optima as a result of *stretching*.

Brits and van den Bergh [14] proposed NichePSO, 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 particle's closest neighbor. The issue of specifying several user parameters still remains. The authors also proposed *nbest* PSO in [37], where a particle's neighborhood best is defined as the average of the positions of all particles in its neighborhood. By computing the Euclidean distances between particles, the neighborhood 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.

Another niching PSO algorithm inspired by the idea of species, Speciation-based PSO (SPSO), was proposed in [15], [16]. A procedure for determining species and the dominant particles in these species was adopted from [13]. Each species and its corresponding species seed (i.e., the dominant particle) form a separate subpopulation that can be run as a PSO itself. Since species are adaptively formed around different optima, over successive iterations multiple global optima can be found in parallel. 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. Recently, two PSO niching algorithms aiming to improve the robustness to such a niching parameter were proposed in [32], [33]. In [32], population statistics were used to adaptively determine the niching parameters during a run, whereas in [33], a time-based convergence measure was used to directly enhance SPSOs' robustness to the niche radius value. These extensions to SPSO made it more robust; however, the need to specify niching parameters (such as the niche radius) remains.

A PSO based on fitness-Euclidean distance ratio (FER-PSO) was recently proposed for multimodal optimization [38]. 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 toward a fittest-and-closest neighborhood point that is identified via computing its FER value. 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).

In [39], a vector-based PSO was developed to combat the issue of having to specify parameters. This algorithm depends on a niche radius defined by the distance from the current particle to the closest particle that moves in an opposite direction. The distance calculation can be expensive since every particle has to be compared with all others, therefore the complexity is $\mathcal{O}(N^2)$. In addition, only simple 1 or 2-D test functions were used in this paper.

B. Can A More Restrictive Communication Topology Help?

It has been observed by many that different communication topologies can be used to control the speed of information propagation in an EA population. More restrictive communication topologies such as ring, star, or von Neumann have been shown to be effective in slowing down the convergence speed of an EA, hence alleviating the problem of premature convergence. However, it was also noted by some that such restrictive topologies cannot induce stable niching bebaviours, as eventually the population would still converge to a single solution in the presence of multiple equally good solutions. For example in [40], it was observed that spatially structured EAs (fine-grained or coarse-grained), where individuals of a population are mapped onto some communication topology, often help maintain a better population diversity, but they did not prevent the population from converging to a single optimum eventually. Similarly, mating restriction, where mating among similar individuals in the population is restricted, though can produce temporary species, but competition between species will eventually eliminate all but one best-fit species. As a result, mating restriction is not effective in maintaining multiple subpopulations [41]–[43].

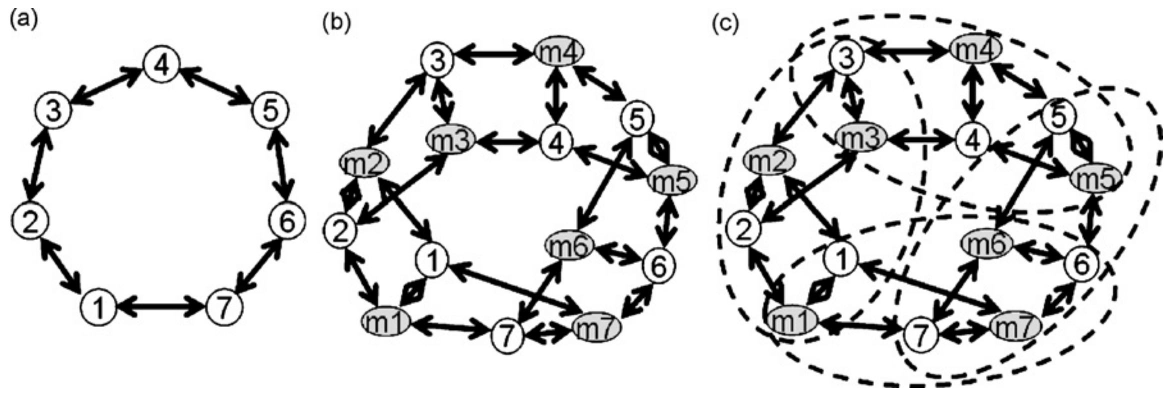


Fig. 2. (a) Ring topology used in a conventional EA. Each member interacts only with its immediate left and right neighbors, with no *local memory* used. (b) Graph of influence for a *lbest* PSO using the same ring topology [[21], p.89]. Each particle possesses a *local memory*. (c) The same as (b) but also showing the overlapping subpopulations, each consisting of a particle and its two immediate neighbors, and their corresponding memories.

In the empirical studies carried out in [18], [24], it was also found that *lbest* PSO algorithms with restricted local neighborhood interactions were unable to form stable niches or were very inefficient in doing so.

We will demonstrate in the following sections that an *lbest* PSO using a ring topology, in fact, is able to induce stable niching behaviors. One possible explanation of the traditional EAs' inability to induce niching behavior is that these EAs do not allow individuals to have *memory*, more specifically *local memory*. In contrast, a *lbest* PSO is able to do so because each particle possesses *local memory* (via its personal best).

C. Memory-Swarm Versus Explorer-Swarm

In PSO, interactions among particles play an important role in particles' behavior. 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 [21], 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 previous position [see (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 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. 2(a) shows an example of a conventional EA using a ring topology with a population of seven individuals. Fig. 2(b) shows a swarm of seven particles using a ring topology, as illustrated by using a "graph of influence" as suggested by Clerc [21]. 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 three informants, from two neighboring particles' memories and its own memory. Each particle's memory also has 3 informants, from two neighboring particles and the particle itself. In stark contrast, Fig. 2(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*.

V. Niching PSOs using a Ring Topology

In this section, we propose to use a *lbest* PSO for niching. We will demonstrate that even with a simple *lbest* PSO employing a typical ring topology, stable niching behaviors can be induced. Kennedy and Mendes [23] studied PSOs using various population topologies including the ring topology. Among all topologies, Kennedy and Mendes considered the ring topology to be "*the slowest, most indirect communication pattern*," whereas the *gbest* PSO represents the "*the most immediate communication possible*." This paper showed communication topology could be used as an effective means to control the speed of convergence for PSO population in search for a single global optimum. However, using various topologies for niching (i.e., aiming to locate multiple optima) was not investigated in this paper.

As shown in Fig. 2(b), in a *lbest* PSO using a ring topology, each particle interacts only with its immediate neighbors. Clearly the ring topology is desirable for locating multiple optima, because ideally we would like to have individuals to search thoroughly in its local neighborhood 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 behavior.

An implementation of such a *lbest* PSO using a ring topology is provided in Algorithm 1. Note that we can

```
Randomly generate an initial population
repeat
    for $i$ = 1 to Population Size do
        if $fit(\vec{\mathbf{x}}_i) > fit(\vec{\mathbf{p}}_i)$ then $\vec{\mathbf{p}}_i \leftarrow \vec{\mathbf{x}}_i$;
    end
    for $i$ = 1 to Population Size do
        $\vec{\mathbf{p}}_{n,i} \leftarrow$ neighborhoodBest($\vec{\mathbf{p}}_{i-1}, \vec{\mathbf{p}}_i, \vec{\mathbf{p}}_{i+1}$);
    end
    for $i$ = 1 to Population Size do
        Equation (3);
        Equation (2);
    end
until termination criterion is met ;
```

Algorithm 1: The pseudocode of a *lbest* PSO using a ring topology.

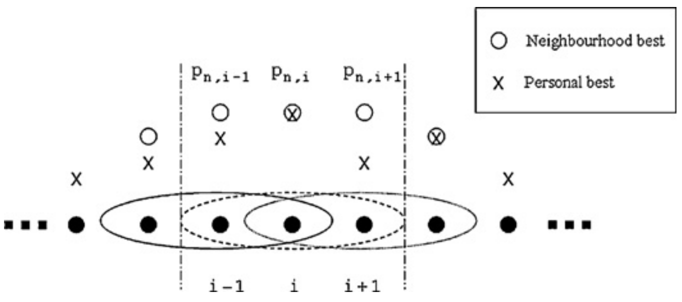


Fig. 3. Ring topology with each member interacting with its two immediate neighbors (left and right). Local neighborhoods are overlapped with each other. The ith particle's neighborhood best ($\vec{\mathbf{p}}_{n,i}$) is the same as those of its two immediate neighboring particles, but differs from those particles in the neighborhoods further out.

conveniently use population indices to identify the left and right neighbors of each particle. Here we assume a "wrap-around" ring topology, i.e., the first particle is the neighbor of the last particle and vice versa. The function *neighborhoodBest*(.) returns the best-fit personal best in the ith neighborhood, which is stored in $\vec{\mathbf{p}}_{n,i}$, representing the neighborhood best for the ith particle. Equation (1) can now be rewritten as the following:

$$\vec{\mathbf{v}}_i \leftarrow \chi(\vec{\mathbf{v}}_i + \vec{\mathbf{R}}_1[0,\varphi_1] \otimes (\vec{\mathbf{p}}_i - \vec{\mathbf{x}}_i) + \vec{\mathbf{R}}_2[0,\varphi_2] \otimes (\vec{\mathbf{p}}_{n,i} - \vec{\mathbf{x}}_i)) \quad (3)$$

where $\vec{\mathbf{p}}_g$ in (1) has been replaced by $\vec{\mathbf{p}}_{n,i}$. The position of the ith particle is now updated according (2) and (3). $\vec{\mathbf{p}}_{n,i}$ is now used as the *local leader* for the ith particle.

Different particles residing on the ring can have different $\vec{\mathbf{p}}_n$ (note that we use $\vec{\mathbf{p}}_n$ to denote a nonspecific "neighborhood best" here), and they do not necessarily converge into a single point over time. As illustrated in Fig. 3, the ring topology not only provides a mechanism to slow down information propagation in the particle population, but also allows different neighborhood bests to *coexist* (rather than becoming homogeneous) over time. This is because a particle's $\vec{\mathbf{p}}_n$ can only be updated if there is a better personal best in its neighborhood, but not by a better $\vec{\mathbf{p}}_n$ of its neighboring particle.

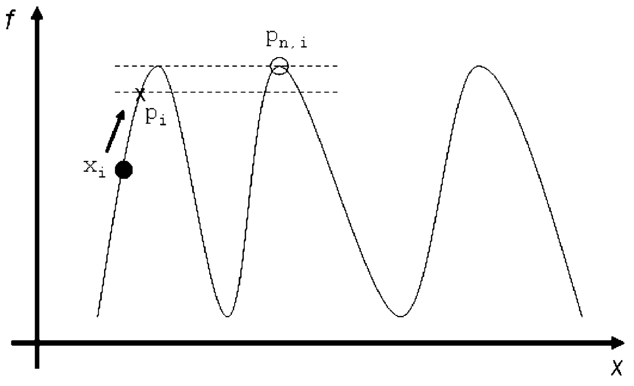


Fig. 4. Example where $\vec{\mathbf{p}}_{n,i}$ and $\vec{\mathbf{p}}_i$ are situated on different peaks.

On a multimodal fitness landscape, a particle's personal best $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_{n,i}$ could be either on the same peak or two different peaks. If they are on the same peak, both $\vec{\mathbf{x}}_i$ and $\vec{\mathbf{p}}_i$ are likely to catch up with the particle's $\vec{\mathbf{p}}_{n,i}$, resulting in $\vec{\mathbf{p}}_i = \vec{\mathbf{p}}_{n,i} = \vec{\mathbf{x}}_i$ (as described in [24]). The particle's velocity $\vec{\mathbf{v}}_i$ will be reduced and eventually the particle stops moving when $\vec{\mathbf{v}}_i$ equals 0. However, if $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_{n,i}$ are sitting on two different peaks, the behavior of the particle's movement is dramatically different. As shown in Fig. 4, the chance of moving this $\vec{\mathbf{p}}_i$ to the same peak as $\vec{\mathbf{p}}_{n,i}$ is in fact very small, since $\vec{\mathbf{x}}_i$ has to move somewhere between the "gap" (as indicated by the dashed lines) and be on the same peak as where $\vec{\mathbf{p}}_{n,i}$ resides. In addition, if $\vec{\mathbf{x}}_i$ moves into the "gap" on the peak where $\vec{\mathbf{p}}_i$ resides, it will only further narrow this gap, resulting in an even smaller chance of moving $\vec{\mathbf{p}}_i$ onto the peak where $\vec{\mathbf{p}}_{n,i}$ resides. Consequently, after some iterations, $\vec{\mathbf{p}}_{n,i}$ and $\vec{\mathbf{p}}_i$ tend to differ for the rest of the run. If this occurs, a scenario like $\vec{\mathbf{p}}_i = \vec{\mathbf{p}}_{n,i} = \vec{\mathbf{x}}_i$ may never arise. According to (2) and (3), $\vec{\mathbf{v}}_i$ will become unlikely to reduce to 0, resulting in the particle oscillating between the $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_{n,i}$, due to the random coefficients from $\vec{\mathbf{R}}_1$ and $\vec{\mathbf{R}}_2$. It is indeed possible that some particles may never fully converge to any point for the entire run. However, the population of $\vec{\mathbf{p}}_n$ and $\vec{\mathbf{p}}_i$ (i.e., memory-swarm) do become stabilized over time, reaching to some equilibrium state.

A. Convergence Behaviors

The convergence behaviors of the ring topology *lbest* PSO can be illustrated by running it on a simple 1-D test function, the *equal maximum* function, which has five evenly spaced global peaks with a height of 1.0 (see f_4 in Table I). For the purpose of illustration, a small swarm of 10 particles was used for the ring topology *lbest* PSO, which was then run for some iterations until all $\vec{\mathbf{p}}_n$ and personal bests stabilized. We employ an index-position plot as a visualization tool to help analyze the niching behavior of the *lbest* PSO. In an index-position plot, the positions of particles are shown in a sequential order as in its ring topology. This way one can observe clearly how niches emerge from interactions among different local neighborhoods. The index-position plot can be used together with a variable-fitness plot to gain valuable firsthand understanding of the niching behavior of the proposed *lbest* PSO niching algorithms.

TABLE I
TEST FUNCTIONS

Name	Test Function	Range	Number of Global Peaks
Two-Peak Trap [45]	$f_1(x) = \begin{cases} \frac{160}{15}(15-x) & \text{for } 0 \leq \text{x} < 15, \\ \frac{200}{5}(x-15) & \text{for } 15 \leq \text{x} \leq 20. \end{cases}$	$0 \leq \text{x} \leq 20$	1
Central Two-Peak Trap [45]	$f_2(x) = \begin{cases} \frac{160}{10}x & \text{for } 0 \leq \text{x} < 10, \\ \frac{160}{5}(15-x) & \text{for } 10 \leq \text{x} < 15, \\ \frac{200}{5}(x-15) & \text{for } 15 \leq \text{x} \leq 20. \end{cases}$	$0 \leq \text{x} \leq 20$	1
Five-Uneven-Peak Trap [13]	$f_3(x) = \begin{cases} 80(2.5-x) & \text{for } 0 \leq \text{x} < 2.5, \\ 64(x-2.5) & \text{for } 2.5 \leq \text{x} < 5.0, \\ 64(7.5-x) & \text{for } 5.0 \leq \text{x} < 7.5, \\ 28(x-7.5) & \text{for } 7.5 \leq \text{x} < 12.5, \\ 28(17.5-x) & \text{for } 12.5 \leq \text{x} < 17.5, \\ 32(x-17.5) & \text{for } 17.5 \leq \text{x} < 22.5, \\ 32(27.5-x) & \text{for } 22.5 \leq \text{x} < 27.5, \\ 80(x-27.5) & \text{for } 27.5 \leq \text{x} \leq 30. \end{cases}$	$0 \leq \text{x} \leq 30$	2
Equal Maxima [42]	$f_4(x) = \sin^6(5\pi x)$.	$0 \leq \text{x} \leq 1$	5
Decreasing Maxima [42]	$f_5(x) = \exp\left[-2\log(2) \cdot \left(\frac{x-0.1}{0.8}\right)^2\right] \cdot \sin^6(5\pi x)$	$0 \leq \text{x} \leq 1$	1
Uneven Maxima [42]	$f_6(x) = \sin^6(5\pi(x^{3/4} - 0.05))$.	$0 \leq \text{x} \leq 1$	5
Uneven Decreasing Maxima [42]	$f_7(x) = \exp\left(-2\log(2) \cdot \left(\frac{x-0.08}{0.854}\right)^2\right) \cdot \sin^6(5\pi(x^{3/4} - 0.05))$.	$0 \leq \text{x} \leq 1$	1
Himmelblau's function [42]	$f_8(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$.	$-6 \leq \text{x} \leq 6$	4
Six-Hump Camel Back [46]	$f_9(x, y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2]$.	$-1.9 \leq x \leq 1.9$; $-1.1 \leq y \leq 1.1$	2
Shekel's foxholes [5]	$f_{10}(x, y) = 500 - \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{1+i+(x-a(i))^6+(y-b(i))^6}}$ where $a(i) = 16(i \bmod 5) - 2)$, and $b(i) = 16(\lfloor(i/5)\rfloor - 2)$	$-65.536 \leq x, y \leq 65.535$	1
Inverted Shubert function [13]	$f_{11}(\vec{\mathbf{x}}) = -\prod_{i=1}^{n} \sum_{j=1}^{5} j\cos[(j+1)x_i + j]$.	$-10 \leq x_i \leq 10$	$n \cdot 3^n$
Inverted Vincent function [31]	$f_{12}(\vec{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} \sin(10 \cdot \log(x_i))$	$0.25 \leq x_i \leq 10$	6^n
Inverted Rastrigin function [47]	$f_{13}(\vec{\mathbf{x}}) = -\sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$.	$-1.5 \leq x_i \leq 1.5$, where $i = 1 \ldots n$	1
Generic Hump function [29]	$f_{14}(\vec{\mathbf{x}}) = \begin{cases} \max_{k=1,K}[h_k(1 - (\frac{d(\vec{\mathbf{x}},k)}{r_k})^{\alpha_k})], & \text{if } d(\vec{\mathbf{x}}, k) \leq r_k; \\ 0 & \text{otherwise.} \end{cases}$	$0 \leq x \leq 1$	Arbitrarily set

Fig. 5(a) shows that in a run with just 10 particles, the *lbest* PSO managed to locate four global peaks at iteration 615 with several distinct $\vec{\mathbf{p}}_n$. From the index-position plot in Fig. 5(b), it is clear that multiple separate niches have established themselves firmly on four different peaks. Sometimes more than one niche could reside on the same peak. A close examination on the raw data in Fig. 6 reveals that six separate niches have formed at iteration 615, with each niche determined by a common $\vec{\mathbf{p}}_n$. These six niches have effectively become six independent PSO optimizers because no interactions are possible across niches by this stage. The six distinct $\vec{\mathbf{p}}_n$ points are able to *coexist*, because each $\vec{\mathbf{p}}_n$ is only determined by its immediate neighbors' personal bests, but not their $\vec{\mathbf{p}}_n$. As an example for Fig. 4, we can see that particle 1s personal best and its $\vec{\mathbf{p}}_n$ are on two different peaks. The fitness "gap" between the personal best and $\vec{\mathbf{p}}_n$ (which is also particle 2s $\vec{\mathbf{p}}_n$) has become so narrow (i.e., less than 0.0000001) that it becomes almost impossible to get an improved personal best for particle 1 on the peak where its $\vec{\mathbf{p}}_n$ resides.

With a population size as small as five, some runs of the ring topology *lbest* PSO were able to reach to an *equilibrium state*, where one or two particles were oscillating around their respective $\vec{\mathbf{p}}_n$, but never converging, as shown in Fig. 5(a).

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 $\vec{\mathbf{p}}_n$ positions being the local attraction points for the particles in the population. With a reasonably large population size, such a *lbest* PSO is able to form stable niches around the identified neighborhood bests $\vec{\mathbf{p}}_n$. As described in Section IV-C, the memory-swarm acts as the *anchor* points retaining the best positions the particles come across, whereas the explorer-

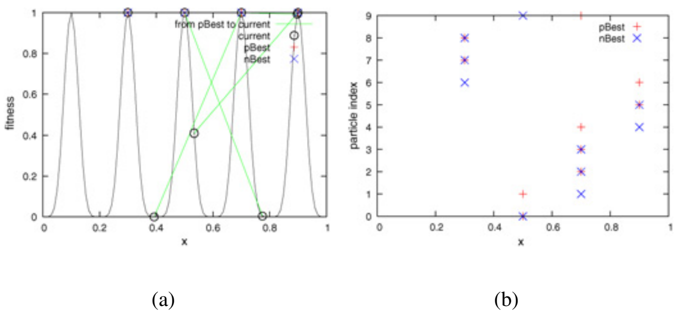
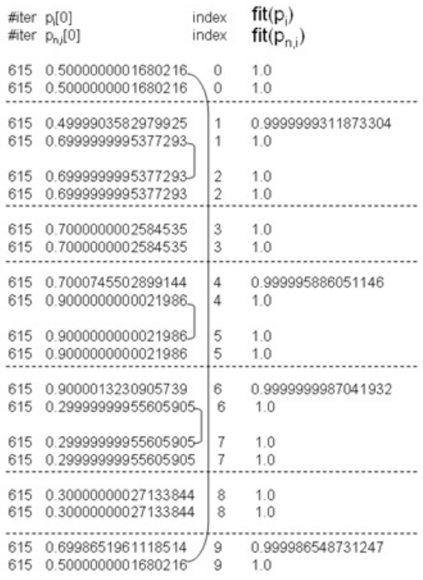


Fig. 5. Ring topology *lbest* PSO on the equal maxima function f_4 at iteration 615. (a) variable-fitness plot. (b) Index-position plot. Note that *nBest* denotes for $\vec{\mathbf{p}}_n$, *pBest* for $\vec{\mathbf{p}}_i$, and *current* for $\vec{\mathbf{x}}_i$. A line is drawn from $\vec{\mathbf{p}}_i$ and its associated $\vec{\mathbf{x}}_i$.



```
#iter  p_i[0]                 index  fit(p_i)
#iter  p_n,i[0]               index  fit(p_n,i)

615  0.5000000001680216      0     1.0
615  0.5000000001680216      0     1.0
------------------------------------------------------------
615  0.4999903582979925      1     0.9999999311873304
615  0.6999999995377293      1     1.0

615  0.6999999995377293      2     1.0
615  0.6999999995377293      2     1.0
------------------------------------------------------------
615  0.7000000002584535      3     1.0
615  0.7000000002584535      3     1.0
------------------------------------------------------------
615  0.7000745502899144      4     0.999995886051146
615  0.9000000000021986      4     1.0

615  0.9000000000021986      5     1.0
615  0.9000000000021986      5     1.0
------------------------------------------------------------
615  0.9000013230905739      6     0.9999999987041932
615  0.29999999955605905     6     1.0

615  0.29999999955605905     7     1.0
615  0.29999999955605905     7     1.0
------------------------------------------------------------
615  0.30000000027133844     8     1.0
615  0.30000000027133844     8     1.0
------------------------------------------------------------
615  0.6998651961118514      9     0.999986548731247
615  0.5000000001680216      9     1.0
```

Fig. 6. Niches formed by a ring topology *lbest* PSO with a population of 10 particles on the equal maxima function at iteration 615. Dash lines separate the stabilized niches that have no further interaction with outside. Each curved line shows a personal best is chosen as the $\vec{\mathbf{p}}_n$ shared by two personal bests on two different peaks. Note that particle 0 is an immediate neighbor of particle 9 since the ring-topology is a "wrap-around."

swarm explores the search space more broadly. One possible explanation on why some classical EAs are unable to form niches around multiple optima (as discussed in Section IV-B) is that they do not have such a memory mechanism to retain equally good search points in parallel, and a mechanism to allow these good points to *coexist*. As a result, selection is biased toward a single dominant individual in the population. In contrast, the *lbest* PSO allows multiple similarly dominant individuals to coexist through its memory-swarm and the scheme for choosing neighborhood bests.

B. Other Possible Variants

In addition to the standard ring topology where each local neighborhood has three members as shown in Fig. 2, some variants can be implemented to achieve similar niching effects as well. One variant is to further restrict the local neighborhood to just two members, as shown in Fig. 7.

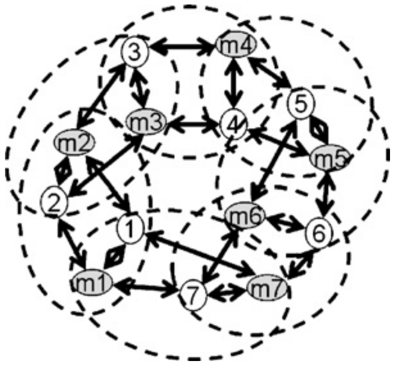


Fig. 7. Graph of influence for a *lbest* PSO with a ring topology using just two members in each local neighborhood.

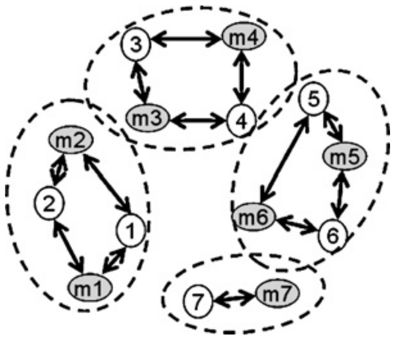


Fig. 8. Graph of influence for a *lbest* PSO. Every two particles form a pair as a local hill-climber. Note that there is no overlapping between neighborhoods.

Since the *lbest* PSO with a ring topology has overlapping neighborhoods, it is still possible for a more dominant $\vec{\mathbf{p}}_n$ to overtake less dominant $\vec{\mathbf{p}}_n$. This leads a tendency of the algorithm locating more dominant peaks over other less dominant peaks. If the goal of optimization is to locate both global and local peaks, then such influence from a more dominant $\vec{\mathbf{p}}_n$ could be further prevented. This can be easily achieved by removing the overlapping neighborhoods in the ring topology. As shown in Fig. 8, each disconnected local neighborhood acts as an independent hill climber, searching for a local optimum only.

Apart from simplicity, these *lbest* PSOs do not require any prior knowledge of (neither the need to specify) any niching parameters, e.g., a niche radius, because niches emerge naturally from their initial population. The complexity of the algorithm is only $\mathcal{O}(N)$ (where N is the population size), as the calculation to obtain a neighborhood best is only done locally from each particle's local neighborhood.

In the following sections, we will demonstrate through extensive experiments that these *lbest* PSOs are able to induce stable niching behaviors, thereby locating multiple global optima reliably for multimodal optimization problems.

VI. Experiments

To evaluate the niching ability of the proposed *lbest* PSOs with a ring topology, we used some widely used multimodal optimization test functions of different characteristics, such as deceptiveness, multiple evenly and unevenly spaced global optima, multiple global optima in the presence of multiple local optima, function rotation, and high dimensionality (see Section VI-A). Our experiments were carried out with the following key aims.

1) To demonstrate the *lbest* PSOs using a ring topology with overlapping neighborhood can stably and reliably locate multiple global optima. In addition, to show that *lbest* PSOs with nonoverlapping neighborhood can locate both global and local optima for some test functions.
2) To compare the *lbest* PSOs using a ring topology with a typical niching algorithm (e.g., SPSO) that does rely on some user parameters such as the niche radius.
3) To study the effect of varying population size, since this is a parameter that a user still needs to supply.

We continue to use the index-position plot as described in Section V-A to illustrate the niching behaviors of the algorithms. Following ring topology, *lbest* PSO variants were used in our experiments.

1) $r3pso$: A *lbest* PSO with a ring topology, each member interacts with its immediate member on its left and right.
2) $r2pso$: A *lbest* PSO with a ring topology, each member interacts with only its immediate member to its right.
3) $r3pso-lhc$: The same as $r3pso$, but with no overlapping neighborhoods. Basically multiple PSOs search in parallel, like local hill climbers. This variant is more appropriate if the goal of optimization is to find global optima as well as local optima.
4) $r2pso-lhc$: The same as $r2pso$, but with no overlapping neighborhoods, hence acting as multiple local hill climbers, more suitable for finding global as well as local optima.

The above *lbest* PSO niching variants were compared with two existing PSO niching algorithms.

1) SPSO: This algorithm was chosen because it represents a state-of-the-art niching algorithm that relies on a prespecified niche radius value [16]. SPSO was inspired by the classic clearing [12], and speciation methods [13], and showed superior performance compared with NichePSO [14] and SNGA [8].
2) FER-PSO: This algorithm was chosen since it shows competitive performance on challenging functions such as the Shubert function.

In addition, in some experiments, we also used the *gbest* PSO, i.e., a standard *gbest* PSO using a global communication topology.

For any particle with a position x_i exceeding the boundary of the variable range, its position is reset to a value that is twice of the right (or left boundary) subtracting x_i.

In the following sections, we first describe the test functions used. We then describe the performance measures used for this paper. For SPSO, since we know the global optima for all the test functions, we always set the niche radius r to a value that is less than the distance between two closest global optima, making sure that it is able to distinguish global optima. In some sense, SPSO was given an "unfair" advantage over the *lbest* PSO algorithms. For all PSO algorithms, we used the standard constricted version as described in Section IV. There was no parameter tuning.

A. Test Functions

Table I shows the test functions used in this paper. These test functions are categorized into six groups ranging from simple to more complex and challenging.

1) *1-D Deceptive Functions:* f_1, f_2, and f_3 are considered to be deceptive. These functions may be difficult because the existing local optima can misguide the population to move away from the true global optimum. For example, in f_1, since 3/4 of the initial population have the values between 1 and 15, offspring generated from these individuals are likely to move toward the local peak at $x = 0$, rather than the global peak at $x = 20$.

f_2 is simply a variation of f_1. Both f_1 and f_2 have only a single global peak. However, f_3 has three local peaks and two global peaks, presenting additional challenge for an optimizer to find both global peaks.

These functions are useful for testing an optimizer's ability to handle *deceptiveness*, i.e., avoiding misguidance from those local peaks (or false global peaks).

2) *1-D Multimodal Functions:* f_4 has five evenly spaced global peaks. f_5 is similar to f_4. The only difference is that the five peaks decrease in height exponentially (so there is only one global peak actually). f_6 is also like f_4, except that now the five global peaks are unevenly spaced. f_7 is like f_6, but with five peaks decrease in height exponentially.

These functions can be used to test a niching algorithm's ability to form niches on multiple peaks (either local or global). In addition, the feature of unevenly spaced peaks is useful to test those niching methods that rely on a user to specify a niche radius parameter.

3) *2-D Multimodal Functions:* f_8 has four global peaks with two closer to each other than the other two. There are no local peaks. f_9 has two global peaks as well as two local peaks. f_{10} has 16 evenly spaced peaks of unequal heights, with one being the global peak. If the goal of optimization is to find all global and local optima, then f_{10} can be used to test a niching EAs ability to locate all 16 peaks.

4) *More Challenging Two or Higher Multimodal Functions:* Both the inverted Shubert function f_{11} and the inverted Vincent function f_{12} are especially interesting in this paper. For f_{11} the inverted Shubert 2-D, there are 18 global peaks in 9 pairs, with each pair very close to each other, but the distance between any pair is much greater [see Fig. 1(a)]. There are 760 local peaks. As the dimensionality increases, the number of global and local peaks also increase quickly. For the n-dimensional inverted Shubert function, there are $n \cdot 3^n$ global peaks unevenly distributed. These global peaks are divided into 3^n groups, with each group having n global peaks being close to each other. Hence for f_{11} Shubert 3-D, there are 81 global peaks in 27 groups; whereas for f_{11} Shubert 4-D, there are 324 global peaks in 81 groups. f_{11} will pose a serious challenge to any niching algorithm relying on a fixed niche radius parameter.

f_{12} the inverted Vincent function has 6^n global peaks, but unlike the regular distances between global peaks in f_{11}, in

f_{12} global peaks have vastly different spacing between them. Furthermore, f_{12} has no local peaks.

5) *Inverted Rastrigin Function:* The inverted Rastrigin function f_{13} can be used to test an optimizer's ability to locate the single global peak in the presence of many local peaks when the problem dimension is increased dramatically. Furthermore, to test niching algorithms' ability to handle non-separable problems, f_{13} is also rotated to introduce parameter interactions between variables, thereby making the function nonseparable. Rotations are performed in the decision space, on each plane using a random uniform rotation matrix [44]. All niching algorithms are run 50 times. A new random uniform rotation matrix is generated for each run of each algorithm for the purpose of an unbiased assessment.

6) *Generic Hump Functions:* To further evaluate the efficacy of the *lbest* PSO niching algorithms on high dimensional problems, the *generic hump function* introduced in [29] was modified and used for experimentation. One major advantage of using the hump function is that unlike the Shubert or Vincent function, it allows an arbitrary number of peaks generated regardless of the number of dimensions. The basic idea is as follows: all variables are initialized within [0, 1]. Within this range, K peaks are generated at random locations, with different shapes and sizes. The radius of each peak (i.e., the radius of the basin of attraction of each maximum) is also randomly created, but the distance between any two neighboring peaks is at least equal to or greater than the radius of one of the two peaks. For the kth peak, its height h_k and shape factor α_k are also randomly chosen. The fitness of a solution $\vec{\mathbf{x}}$ is calculated in the following steps: 1) first identify all peaks that $\vec{\mathbf{x}}$ reside on; 2) calculate the Euclidean distances between $\vec{\mathbf{x}}$ and the centers of these peaks; and 3) calculate the fitness of $\vec{\mathbf{x}}$ according to

$$f_{14}(\vec{\mathbf{x}}) = \begin{cases} \max_{k=1,K}[h_k(1 - (\frac{d(\vec{\mathbf{x}},k)}{r_k})^{\alpha_k})], & \text{if } d(\vec{\mathbf{x}}, k) \le r_k; \\ 0 & \text{otherwise} \end{cases}$$

where r_k denotes the radius of the kth peak, and $d(\vec{\mathbf{x}}, k)$ denotes the Euclidean distance between $\vec{\mathbf{x}}$ and the center of the kth peak. The original hump function in [29] does not allow peaks to intersect, hence producing too much flat surface between peaks. Consequently, the fitness values of many individuals in the initial population are likely to be zero. The above modified hump function alleviates such a problem by allowing multiple peaks to intersect with each other. The fitness is simply the maximal height value of an individual on all the peaks it resides on. By choosing different values for h_k, r_k, α_k and the maximal number of peaks K, multimodal test functions having different complexities can be created.

B. Population Size and Maximal Number of Evaluations

For functions f_1 to f_{10}, we used a population size of 20–50, which should be adequate for locating all global peaks. All PSO niching algorithms were run (for f_1 to f_{10}) for a maximum of 10 000 function evaluations, or having located all known global peaks with the specified accuracy. For f_{11} Shubert 2-D function, population sizes ranging from 200 to 500 were used, since it has many more global peaks (i.e., 18 global peaks). We also showed the effect of varying population sizes using f_{11} Shubert 2-D. Following the same principle, larger population sizes up to 1000 were used for f_{11} Shubert 3-D and 4-D, f_{12} Vincent 1-D to 4-D and f_{13} Rastrigin 2-D to 15-D.

For more challenging test functions, the maximal number of evaluations allowed was increased accordingly. For f_{11} 4-D, the maximal number of evaluations was 400 000. For f_{12} Vincent 2-D, 3-D, and f_{13} Rastrigin 2-D to 15-D, the maximal number of evaluations was 200 000.

For f_{14} the generic hump function we used a population size from 300 to 800 for dimensions ranging from 8 to 20, respectively. Ten peaks were created ($K = 10$) for all dimensions. The maximal number of evaluations was set to 200 000.

C. Setting Niche Radius for Higher Dimensional Functions

For higher dimensional multimodal functions, Deb and Goldberg [43] proposed a method to compute the value of the niche radius r in a D-dimensional space where there exist p global optima

$$r = \frac{\sqrt{\sum_{k=1}^{D}(x_k^u - x_k^l)^2}}{2\sqrt[D]{p}} \tag{4}$$

where x_k^l and x_k^u are the lower and upper bounds on the kth dimension of the variable vector of D dimensions. This method, however, assumes that the number of global optima is known and they are evenly distributed in the search space. As can be seen from Fig. 1, both f_{11} Shubert and f_{12} Vincent functions have many unevenly distributed global optima. Hence, (4) cannot be applied to obtain an estimated niche radius for a typical niching method relying on a fixed niche radius r, e.g., SPSO. In contrast, *lbest* niching PSOs do not rely on any niche radius value, hence they certainly provide a distinct advantage over SPSO.

Due to the difficulty in applying (4), for 2-D functions SPSO was provided with a niche radius value normally set to be less than the distance between two closest global optima. However, for higher dimensional functions with uneven distributions of global optima, too small niche radius is likely to create too many small niches, resulting in premature convergence. Assuming that no prior knowledge about the distribution of global peaks is readily available, for higher dimensional functions, we simply used the same niche radius values, which were empirically found optimal for the 2-D function counterparts.

D. Performance Measures

To compare the performance of the proposed *lbest* niching PSOs with those of SPSO and FER-PSO, we first allow a user to specify a *level of accuracy* (typically $0 < \epsilon \le 1$), i.e., how close the computed solutions to the known global peaks are. If the distance from a computed solution to a known global optimum is below the specified ϵ, then we can consider the peak is found. For only the purpose of measuring performance, we make use of an algorithm for identifying species seeds [16], in order to check if a niching algorithm has located all known global peaks. Basically at the end of each run, this algorithm is invoked to first sort all individuals in the population in

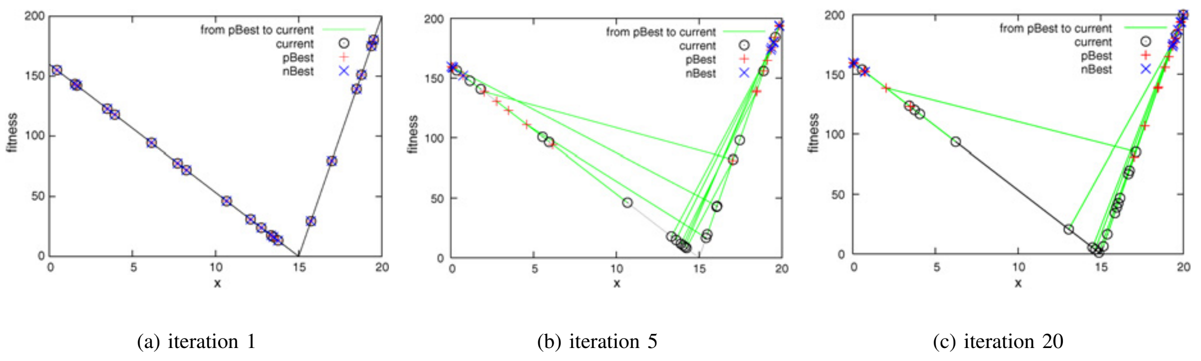


Fig. 9. Niching behavior induced by $r3pso$ on the f_1. Majority of the 20 particle personal bests gradually move toward the global optimum.

decreasing order of fitness values. With a prespecified niche radius, we iterate from the best-fit individual on the sorted list, to check if any other individuals are within the niche radius from it. If so, they are tagged as belonging to the same species. These individuals are then excluded from the sorted list. The next best-fit individual on the sorted list is then considered, and the above process is repeated. The algorithm terminates when there is no individual left.

As long as the niche radius r is set to a value not greater than the distance between two closest global peaks, individuals on two found global peaks would be treated as from different species. The species seeds identification algorithm will produce a list of best as well as different personal best positions based on the prespecified niche radius and a given list of all personal best positions from the swarm population. For the test functions in Table I, since the exact number of global peaks is known *a priori*, and also roughly how far apart between two closest global peaks, a niching algorithm's performance can be measured in terms of the *number of evaluations* required to achieve a given accuracy ϵ for all known global peaks for a run. In this case, we only need to check species seeds, which are the dominant particles sufficiently different from each other. We can determine if a global peak is found by checking each species seed to see if it is close enough to a known different global peak. Note again that this species seeds identification algorithm is only used for performance measurement in determining if a sufficient number of global peaks has been found, but not in any part of the optimization procedure.

For real-world problems, it is not uncommon that the number of global optima is unknown. However, the above species seeds identification algorithm is still useful. A practitioner can still run a niching algorithm for a certain number of iterations before invoking this algorithm (with a reasonably small niche radius value). Presuming that the best-fit individual has found a global optimum (this is often the case for low dimensional problems), then seeds on the sorted list can be checked against this best-fit seed to see if they are good enough in fitness as well as sufficiently different.

The performance of all compared PSO niching algorithms is measured in terms of the *success rate*, which is the percentage of runs in which all global peaks are successfully located. Note that the success rate can depend on the specified ϵ. For a more relaxed (or higher value) of ϵ, an algorithm is more likely to have a higher success rate.

For more challenging functions, it is possible that *success rate* may become 0 if not all peaks are found in any run. In such a case, the number of global peaks found in a run is recorded, and averaged over 50 runs.

VII. Numerical Results

This section summarizes the results and analysis on the experiments carried out.

A. 1-D Deceptive Functions

1) f_1: For f_1, if ϵ is set to 0.1, i.e., $r3pso$ has 100% success rate in locating the global peak. However, if ϵ is set to 0.0001, the success rate drops to 94%. This shows that the fine-tuning ability of an *lbest* PSO such as $r3pso$ is not so good, but it has always managed to find at least some points in a very close vicinity of the global peak ($x = 20$).

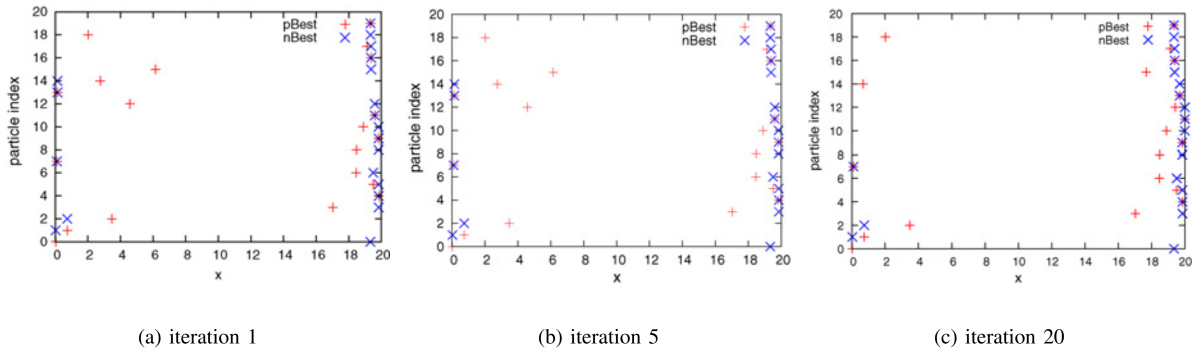
Fig. 9 shows a simulation run of the $r3pso$ on f_1 using only 20 particles. In just 20 iterations, the population managed to find the true global optimum. Fig. 10 shows that most $\vec{\mathbf{p}}_n$ have moved to the vicinity of the global peak, though the catchment area of the global peak is only 1/3 of that of the local peak.

Our experiment shows that even the *gbest* PSO performed remarkably well on f_1. With a population size of 50, *gbest* PSO always located the true global optimum, instead of being attracted to the local peak. If it runs long enough, all particles will end up on the slope of the global peak (Fig. 11). This contrasts with the fact that often a simple GA is being misled to the local peak due to the 75% of the initial population fall on the slope leading up to the local peak. The *gbest* PSO did not lose good solutions found largely due to its use of *personal bests*, but a GA does not have such a mechanism.

With a respectable ϵ set to 0.1, $r3pso$, $r2pso$, *gbest* PSO, and SPSO were all able to locate the true global optimum with 100% success rate. SPSO was the fastest in locating the global peak. However, SPSO requires a user to supply a niche radius value.

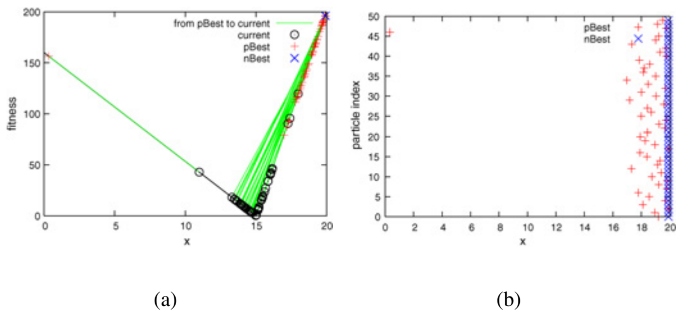
If $r3pso-lhc$ is used, then both the global and local optima will be located with 100% success rate.

The results on f_2 are similar to those of f_1, hence we will not show them here.



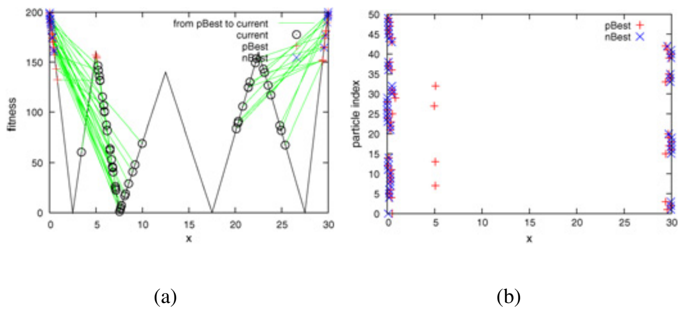
(a) iteration 1 (b) iteration 5 (c) iteration 20

Fig. 10. Index-position plot for the same run as given in Fig. 9 shows that most of the niches are formed around the global optimum.



(a) (b)

Fig. 11. *gbest* PSO on the two-peak trap function at iteration 60. (a) Variable-fitness plot. (b) Index-position plot.

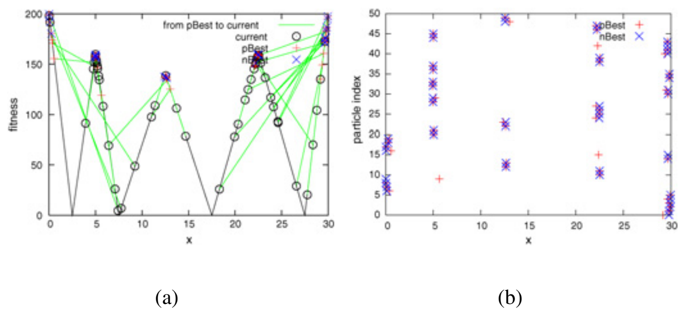


(a) (b)

Fig. 12. $r3pso$ on the five-uneven-peak-trap function f_3 at iteration 100 (with a population size of 50). (a) Variable-fitness plot. (b) Index-position plot.

2) f_3: f_3 has five peaks, of which two are global peaks and three are local peaks. It presents additional challenges than f_1 and f_2 as two global peaks must be located, while avoiding the three local peaks. Fig. 12(a) shows that $r3pso$ can locate both global peaks without any difficulty. It is interesting to see that all $\vec{\mathbf{p}}_n$ are located on the two global peaks (i.e., none on local peaks), which helps the population remain on the two global peaks stably. However, particles on local peaks are unstable, as their corresponding $\vec{\mathbf{p}}_n$ on the slopes of two global peaks continue to attract them to move toward the two global peaks. Fig. 12(b) shows clearly that there are three or four stable niches formed around each global peak.

Local hill climbers such as $r2pso - lhc$ and $r3pso - lhc$ were shown to be more effective in locating both global and



(a) (b)

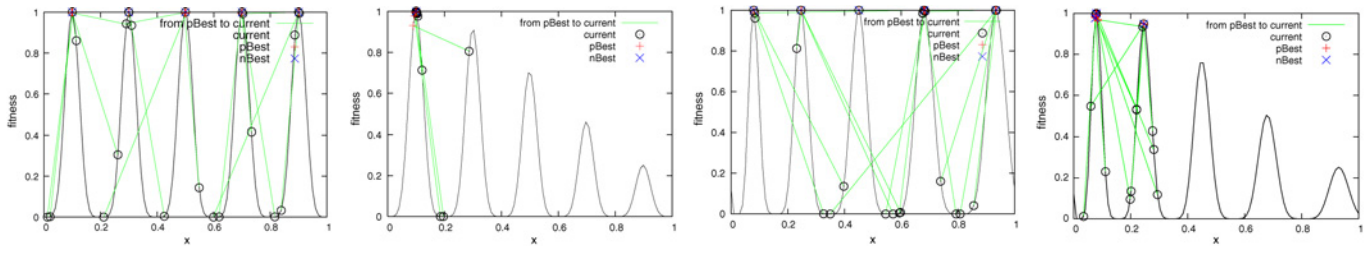
Fig. 13. $r2pso - lhc$ on the five-uneven-peak-trap function f_3 at iteration 20 (with a population size of 50). (a) Variable-fitness plot. (b) Index-position plot.

local peaks. Fig. 13 shows that $r2pso - lhc$ was able to locate all global and local peaks. Since $r2pso - lhc$ has only two members for each neighborhood, many small niches were formed around the five peaks. Each niche is in fact a local hill climber, i.e., an independent PSO with a population size of just two.

B. 1-D Multimodal Functions

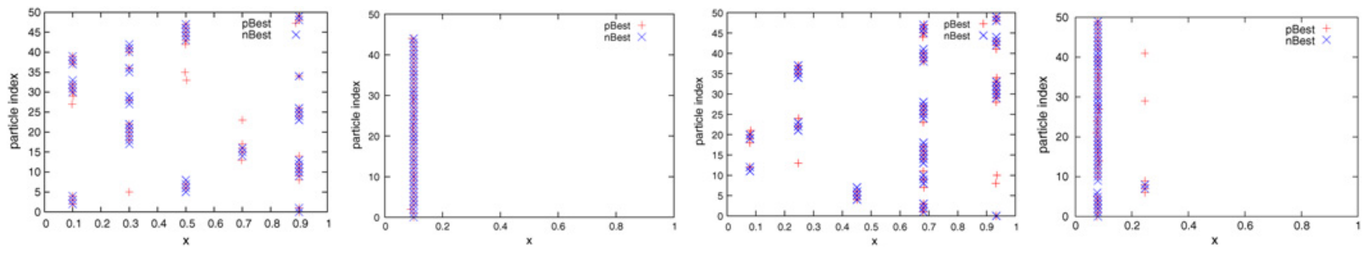
f_4, f_5, f_6, and f_7 were introduced by Deb [42] for testing his sharing GA. f_4 and f_6 both have five global peaks. Most interesting is that f_6s peaks are unevenly spaced. As a result, this would require any niching method that relies on a uniform niche radius, to choose an appropriate niche radius value. Choosing a value either too large or small will cause the niching method either being unable to distinguish between two peaks, or prematurely converged. Figs. 14 and 15 show that $r3pso$ had no difficulty in locating all global peaks on all Deb's four functions. For f_5 and f_7, particles were attracted toward the single global peak because it has the best-fit $\vec{\mathbf{p}}_n$. Since $r3pso$ has overlapping neighborhoods, the influence from the best-fit $\vec{\mathbf{p}}_n$ on the global peak eventually took over the population.

For locating not only the global peak, but also the local peaks in f_5 and f_7, local hill climbers $r2pso - lhc$ and $r3pso - lhc$ performed consistently well. Since each niche performs searches independently of each other, particles that have located lower peaks are not influenced by particles on the more dominant peaks. Fig. 16 shows that $r2pso - lhc$



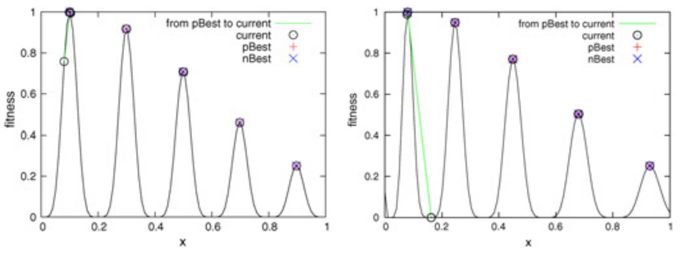
(a) f_4 (b) f_5 (c) f_6 (d) f_7

Fig. 14. $r3pso$ (with a population size of 50) on the Deb's four functions (f_4 to f_7) after 100 iterations.



(a) f_4 (b) f_5 (c) f_6 (d) f_7

Fig. 15. Index-position plot for each function shown in Fig. 14.



(a) f_5 (b) f_7

Fig. 16. $r2pso - lhc$ on f_5 and f_7 respectively, after 100 iterations (with a population size of 50).

performed well on these 2 functions. $r2pso - lhc$ did not have any difficulty with the uneven distances between the peaks.

C. 2-D Multimodal Functions, f_8 to f_8

The results on f_8 Himmelblau suggest that f_8 favors more $r2pso - lhc$, $r3pso - lhc$, FER-PSO, and SPSO than the more connected $r2pso$ and $r3pso$. Since two out of the four global peaks are very close to each other, for $r2pso$ and $r3pso$ there is a tendency for particles on these two peaks to merge into one niche. This suggests that for functions having very close global peaks, more restricted or even independent local hill climbers might be better optimizers. On the other hand, for functions having many local peaks and only a few distant global peaks, in order to escape local peaks, more connected topologies would be better suited. For example, in the case of f_{10} Shekel's foxholes function that has 16 evenly distributed peaks (including one global peak), $r2pso$ and $r3pso$ outperformed their local hill climber counterparts. These results are shown in Table II, which is described in the next section.

D. Success Rates

Table II summarizes the success rates on f_1 to f_{10}. A population size of 50 was used. The PSO niching algorithms were run until all known global peaks were found, or a maximum of 100 000 evaluations were reached. Note that ϵ and r (niche radius) values were chosen in order to maximally measure the ability of each algorithm in forming niches in the vicinities of all known global peaks. The uses of ϵ and r were purely for performance measurements. Table II shows that all ring topology based *lbest* PSO algorithms and FER-PSO give comparable or better performance than SPSO. Most noticeably is that SPSO gives inconsistent performances across different functions, with very poor performance on f_1, f_2, f_3, and f_{10}, though scoring perfectly for f_4 to f_9. A close look reveals that SPSO has difficulty in handling situations where a global peak is located at the boundary of the search space, such as f_1 to f_3. For f_{10}, SPSO was more inclined to be trapped to some of the 15 local peaks than the ring topology *lbest* PSOs. Table III shows the number of evaluations for each algorithm to achieve the success rates presented in Table II. SPSO is certainly the fastest for f_4 to f_9, but for other functions, the ring topology *lbest* and FER-PSO are better. Two additional factors are not considered here: 1) both SPSO and FER-PSO would have a higher cost on sorting the population;

TABLE II
SUCCESS RATES FOR f_1 TO f_{10} FUNCTIONS

fnc	ϵ	r	r2pso (%)	r3pso (%)	r2pso-lhc (%)	r3pso-lhc (%)	FER-PSO (%)	SPSO (%)
f_1	0.1	0.5	98	**100**	94	78	88	24
f_2	0.1	0.5	**100**	96	98	88	**100**	22
f_3	5	0.5	**100**	96	96	96	98	40
f_4	0.01	0.01	**100**	**100**	**100**	**100**	**100**	**100**
f_5	0.01	0.01	98	**100**	**100**	**100**	**100**	**100**
f_6	0.01	0.01	98	98	**100**	**100**	**100**	**100**
f_7	0.01	0.01	**100**	**100**	**100**	**100**	**100**	**100**
f_8	0.1	0.5	92	74	**100**	98	98	**100**
f_9	0.01	0.5	**100**	**100**	**100**	**100**	**100**	**100**
f_{10}	0.01	0.5	**100**	**100**	72	78	**100**	50

TABLE III
AVERAGED NUMBER OF EVALUATIONS OVER 50 RUNS (MEAN AND ONE STANDARD ERROR) FOR RESULTS PRESENTED IN TABLE VII

fnc	ϵ	r	r2pso	r3pso	r2pso-lhc	r3pso-lhc	FER-PSO	SPSO
f_1	0.1	0.5	3.46E+3(1974.95)	**2.62E+03(874.07)**	7.39E+03(3348.05)	2.32E+04(5834.66)	1.44E+04(4535.92)	7.72E+04(5859.19)
f_2	0.1	0.5	2.96E+03(1520.06)	5.34E+03(2764.73)	4.34E+03(2229.64)	1.31E+04(4588.84)	**2.11E+03(227.71)**	7.83E+04(5856.06)
f_3	5	0.5	**9.78E+02(186.65)**	4.65E+03(2784.21)	4.71E+03(2783.41)	6.73E+03(3088.01)	2.66E+03(1992.22)	6.33E+04(6773.53)
f_4	0.01	0.01	3.76E+02(30.55)	4.43E+02(51.80)	3.96E+02(51.01)	4.47E+02(52.75)	3.84E+02(29.01)	**3.55E+02(30.55)**
f_5	0.01	0.01	2.12E+03(1999.53)	1.41E+02(11.22)	1.43E+02(14.64)	1.44E+02(13.68)	1.70E+02(12.78)	**1.27E+02(9.39)**
f_6	0.01	0.01	2.43E+03(1994.23)	2.44E+03(1994.73)	4.56E+02(33.73)	6.23E+02(273.13)	3.71E+02(31.72)	**3.43E+02(23.91)**
f_7	0.01	0.01	1.75E+02(17.91)	1.60E+02(20.20)	1.78E+02(18.14)	1.62E+02(16.88)	1.89E+02(20.17)	**1.44E+02(13.82)**
f_8	0.1	0.5	7.87E+03(2891.69)	2.14E+04(5467.19)	1.49E+03(138.32)	7.38E+03(3347.13)	5.07E+03(1945.99)	**1.25E+03(45.95)**
f_9	0.01	0.5	**6.19E+02(24.11)**	6.84E+02(30.02)	6.18E+02(30.26)	6.50E+02(25.03)	9.65E+02(53.99)	6.53E+02(32.9)
f_{10}	0.01	0.5	4.36E+03(559.98)	3.51E+03(453.55)	2.97E+04(6277.07)	2.48E+04(5738.45)	**3.47E+03(336.15)**	4.28E+04(6968.96)

and 2) SPSO was tuned with user specified niche radius values, but no such parameter is used for the ring topology *lbest* PSOs.

E. f_{11} 2-D, 3-D, and f_{12} 1-D

For more challenging functions f_{11} inverted Shubert 2-D and 3-D, a population size of 500 was used. And for f_{12} inverted Vincent 1-D, a population size of 100 was used. As Table IV shows that the best overall performer is $r3pso$. Even $r3pso - lhc$ did well. The worst performer is SPSO, even though it was given the knowledge to specify a reasonable niche radius r. Both FER-PSO and SPSO completely failed on f_{11} inverted Shubert 3-D. Table V shows the averaged number of evaluations for the corresponding results in Table IV. Fig. 17 shows that $r3pso$ was able to locate all 18 global peaks on f_{11} inverted Shubert 2-D by iteration 75 in a single run. Multiple emerged niches are clearly visible.

f_{11} inverted Shubert 4-D has 324 global peaks. Even with a large population size, it was becoming difficult to find all peaks in any run. Hence we measured the *number of global peaks found* by each algorithm, instead of using *success rate*. We used a population size of 1000, and ran all algorithms for a maximum of 400 000 evaluations.

For f_{12} inverted Vincent 2-D and 3-D, there are 36 and 216 global peaks, respectively. The distances between these global peaks are vastly different, making them difficult for any niching algorithm relying on a uniform niche radius value. A population size of 500 and 1000 was used for f_{12} 2-D and 3-D, respectively. All niching variants were run for 200 000 evaluations.

Table VI shows the number of global peaks (averaged over 50 runs) found by all PSO niching variants. None of them was able to find all global peaks. All *lbest* PSOs gave much better results than SPSO on f_{11} 4-D, and comparable results on f_{12} 2-D and 3-D. On f_{12} Vincent 2-D, Fig. 18 shows that $r3pso$ was able to develop stable niches on the majority of the global peaks, without much concern to the vastly different distances between these peaks.

F. Inverted Rastrigin Function

f_{13} the inverted Rastrigin function has only a single global peak, and many local peaks. The number of local peaks increases exponentially as the dimension increases. To locate the single global optimum, niching algorithms will have to overcome these local peaks. We carried out the following experiment to study the effect of increasing dimensionality on the performance. We assume that there is no prior knowledge of the number of global peaks and local peaks, and neither the distance between the closest global peaks. The only knowledge we have is the upper and lower bounds of the variables to be optimized. This is the only information we use to set an estimated niche radius value. In this case, we set the niche radius value for SPSO to 5.12, which is half of the distance between the lower and upper bounds [we did not use (4), as it assumes the number of optima is known *a priori*]. Of course, this niche radius r should have no influence on the performance of *lbest* PSOs. We set ϵ to 5, so that we consider an algorithm has located the global peak if the difference between the fitness of the global peak and the best-fit particle is less than 5. Fig. 19 shows the success rates of all niching PSOs

TABLE IV

SUCCESS RATES ON f_{11} INVERTED SHUBERT 2-D AND 3-D, AND ON f_{12} INVERTED VINCENT 1-D

fnc	ϵ	r	**r2pso (%)**	**r3pso (%)**	**r2pso-lhc (%)**	**r3pso-lhc (%)**	**FER-PSO (%)**	**SPSO (%)**
f_{11} (2-D)	0.1	0.5	90	98	98	**100**	56	49
f_{11} (3-D)	0.2	0.5	4	**100**	4	92	0	0
f_{12} (1-D)	0.01	0.2	**94**	86	92	90	88	84

TABLE V

AVERAGED NUMBER OF EVALUATIONS OVER 50 RUNS (MEAN AND ONE STANDARD ERROR) FOR THE RESULTS PRESENTED

fnc	ϵ	r	**r2pso**	**r3pso**	**r2pso-lhc**	**r3pso-lhc**	**FER-PSO**	**SPSO**
f_{11} (2-D)	0.1	0.5	5.59E+04(2676.00)	3.91E+04(1648.14)	3.78E+04(1480.85)	**3.24E+04(581.97)**	9.49E+04(1261.83)	6.16E+04(4463.33)
f_{11} (3-D)	0.2	0.5	1.99E+05(830.26)	**7.40E+04(2343.35)**	1.98E+05(1789.94)	8.13E+04(5849.36)	2.00E+05(0)	2.00E+05(0)
f_{12} (1-D)	0.01	0.2	**8.31E+03(3371.59)**	1.54E+04(4906.90)	9.60E+03(3824.11)	1.47E+04(4344.29)	1.30E+04(4601.37)	1.70E+04(5192.03)

TABLE VI

AVERAGED NUMBER OF GLOBAL PEAK SOLUTIONS FOUND (MEAN AND ONE STANDARD ERROR)

fnc	ϵ	r	**r2pso**	**r3pso**	**r2pso-lhc**	**r3pso-lhc**	**SPSO**
f_{11} (4-D)	0.2	0.5	115.8(1.62)	**185.52(1.25)**	136.78(1.37)	182.6(1.16)	4.5 (0.26)
f_{12} (2-D)	0.01	0.2	25.18(0.3)	22.82(0.25)	25.88(0.31)	24.78(0.36)	**28.6(0.24)**
f_{12} (3-D)	0.01	0.2	76.16(0.65)	66.68(0.66)	**84.18(0.61)**	82.1(0.56)	74.48(0.59)

TABLE VII

AVERAGED NUMBER OF GLOBAL PEAK SOLUTIONS FOUND AND TIME TAKEN FOR THE HUMP FUNCTIONS (MEAN AND ONE STANDARD ERROR)

Dims		**r2pso**	**r3pso**	**r2pso-lhc**	**r3pso-lhc**	**SPSO**
8	No. of peaks found	3.68(0.16)	3.80(0.22)	3.60(0.14)	5.24(0.19)	**5.56(0.20)**
	Time taken	6485.00(246.32)	6457.64(39.67)	5382.40(59.67)	5709.76(21.45)	**11236.88(498.89)**
10	No. of peaks found	2.74(0.19)	3.62(0.21)	0.82(0.13)	3.82(0.18)	**4.10(0.20)**
	Time taken	8921.82(548.39)	8836.20(168.93)	9078.86(346.48)	8202.72(150.62)	**38742.14(2249.15)**
14	No. of peaks found	2.26(0.14)	**2.66(0.16)**	0.00(0.00)	1.62(0.16)	0.66(0.11)
	Time taken	27205.36(2888.24)	19240.34(1690.72)	42029.00(1983.71)	32436.28(2561.68)	**129112.42(2730.24)**
18	No. of peaks found	0.60(0.12)	**1.58(0.15)**	0.00(0.00)	0.04(0.03)	0.00(0.00)
	Time taken	91436.94(3721.06)	80859.68(3138.12)	96248.84(815.83)	83970.94(2202.69)	**198891.58(4121.75)**
20	No. of peaks found	0.16(0.05)	**0.68(0.11)**	0.00(0.00)	0.00(0.00)	0.00(0.00)
	Time taken	150470.64(3352.18)	1144292.68(1898.63)	146460.56(834.42)	143922.56(1347.14)	**293197.86(6064.52)**

over increasing dimensions from 2 to 15. It is noticeable that SPSOs performance degraded more rapidly than $r2pso$ and $r3pso$. The performance of two local hill climbers $r2pso-lhc$ and $r3pso - lhc$ also degraded very quickly, which is not surprising, as they have multiple independent optimizers each consisting of just two and three particles, respectively. However, the better connected $r2pso$ and $r3pso$ fared better. The results suggest that even if the goal is to locate a single global peak in the presence of a massive number of local peaks, and with no prior knowledge of the problem domain, it may be preferable to use *lbest* PSOs using a ring topology, rather than a niching method relying on a fixed niche radius value. Especially $r3pso$ showed a better scalability to increasing dimensions.

The same experiment was repeated on the rotated version of f_{13} inverted Rastrigin function. Comparing Fig. 20 with Fig. 19, it can be noted that all niching algorithms suffered from performance loss as a result of rotation of the function. However, the best performers are still $r3pso$ and $r2pso$. SPSOs performance also suffered badly. As remarked by Clerc [21] and Jason and Middendorf [48], the movements of particles in a standard PSO have a clear dependency on the coordinate axes. There is no exception for PSO niching algorithms. Nevertheless, the ring topology-based *lbest* PSOs performed better than SPSO.

G. Generic Hump Functions Up To 20 Dimensions

In the experiments on f_{14} the generic hump functions, the radius of each peak was randomly chosen, however, the distance between any two neighboring peaks must be at least equal or greater than the radius of one of the two peaks. A constant height $h_k = 1.0$ and a constant shape factor $\alpha_k = 1.0$ were chosen. An algorithm is said to have found a peak if it is able to find a solution at least within 0.1 times the radius of the peak from the peak's midpoint. The niche radius for SPSO was set to 0.5. The population size for 8, 10, 14, 18, and 20 dimensions were set to 300, 400, 500, 600, and 800, respectively.

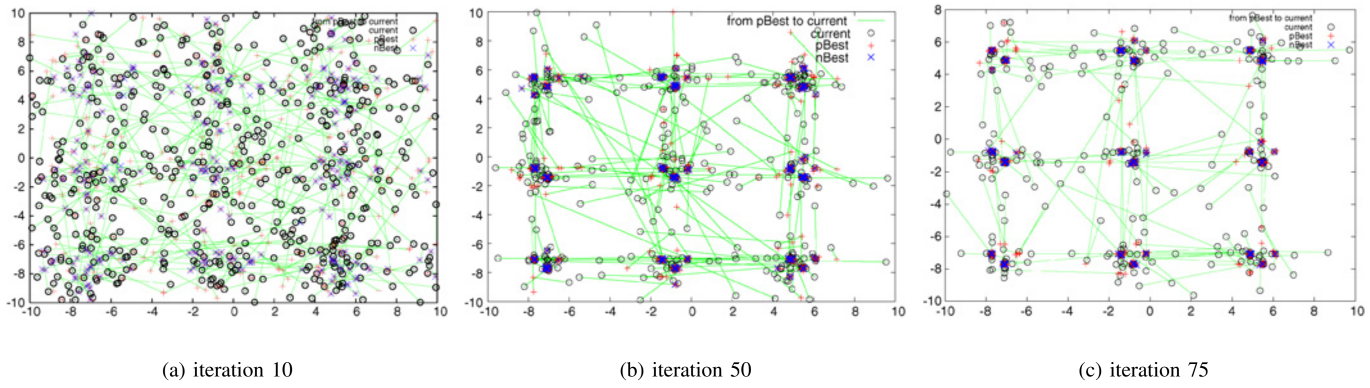


Fig. 17. Niching behavior of the $r3pso$ (with a population size of 500) on the f_{11} Inverted Shubert 2-D function over a run.

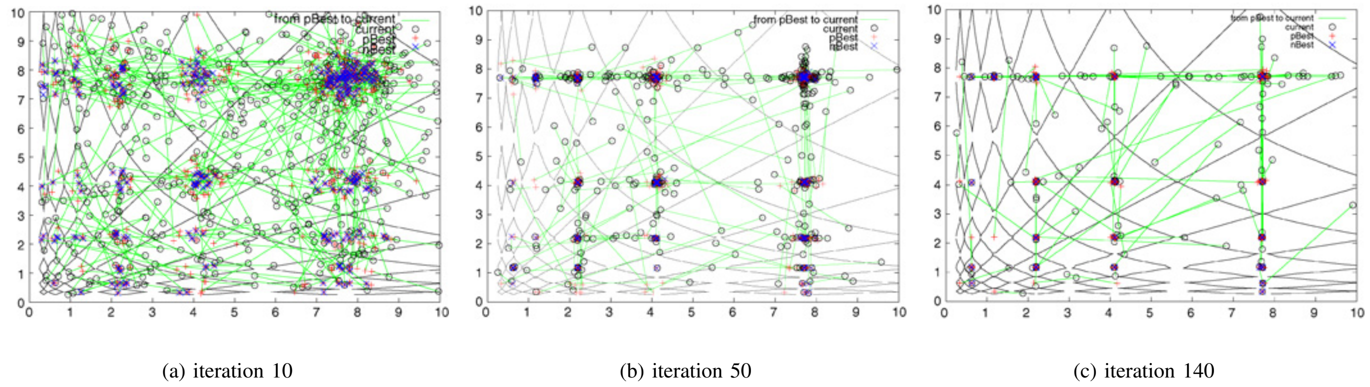


Fig. 18. Niching behavior of the $r3pso$ (with a population size of 500) on f_{12} the inverted Vincent 2-D function over a run.

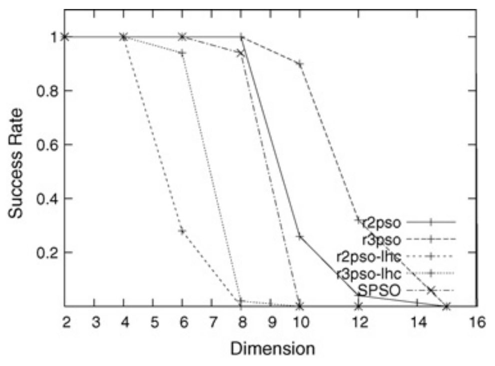


Fig. 19. Success rates for varying dimensions on f_{13} inverted Rastrigin function.

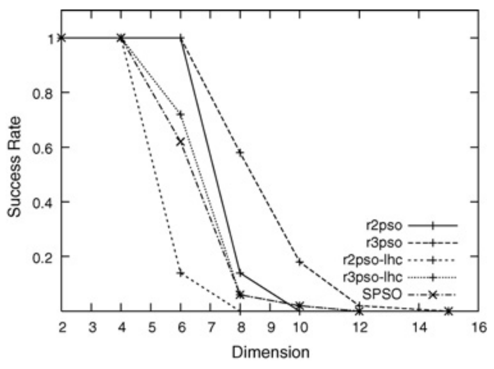


Fig. 20. Success rates for varying dimensions on f_{13} rotated inverted Rastrigin function.

Table VII shows the results on the hump functions. No algorithms were able to find all 10 peaks, hence the number of peaks found was used as the performance indicator. The averaged time taken (i.e., milliseconds) by each algorithm is also included. It can be noted that SPSO performed better than *lbest* ring topology based PSOs on dimensions 8 and 10. However, as the dimension increased further, $r3pso$ became the best performer, which shows $r3pso$ scaled better than SPSO. All algorithms suffered from performance degradation as the dimension increased, among which the local hill climbers $r2pso-lhc$ and $r3pso-lhc$ were the worst, as one would have expected. It is also noted that SPSO took a considerable longer time than *lbest* PSOs, because it had to sort all individuals in a population at each iteration.

H. Maintaining Found Optima

A good niching algorithm should be able to locate global optima and maintain them until the end of a run (see also Section II). All ring topology *lbest* PSO niching algorithms fulfil this requirement, because the *memory-swarm* (in other

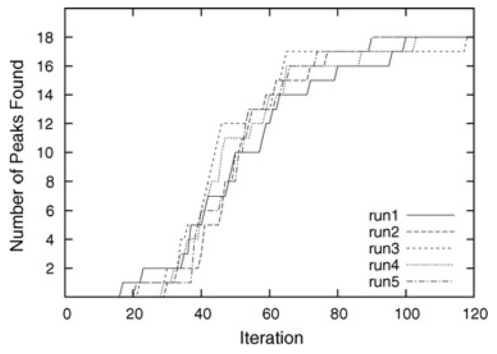


Fig. 21. Number of global optima found by $r3pso$ over five independent runs, on f_{11} inverted Shubert 2-D function (using a population size of 300).

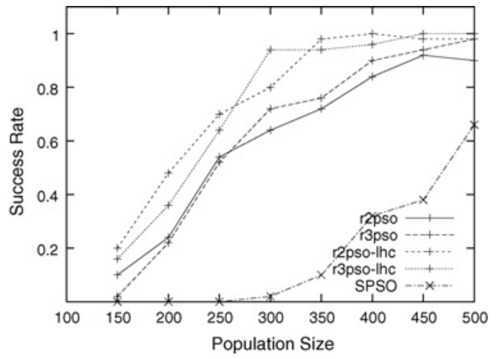


Fig. 22. Success rates for varying population sizes on f_{11} inverted Shubert 2-D function.

words the population of personal bests) forms a stable network retaining the best positions (i.e., personal best positions) found so far by the swarm population. These personal best positions are only updated if their corresponding current positions are better. Otherwise, they remain unchanged. This means a particle will never lose the best position it has found so far. A ring topology *lbest* PSO allows different personal best positions to *coexist* on a multimodal fitness landscape. In contrast, personal best positions in a standard PSO tend to become homogenous eventually converging toward the single global optimum.

Fig. 21 shows five independent runs of $r3pso$ on f_{11} the inverted Shubert 2-D function. For each run, $r3pso$ was able to continuously locate more global peaks and maintain them until all 18 global peaks were found. All ring topology *lbest* PSO niching algorithms share this property; therefore we will not show other results.

I. Effect of Varying Population Size

For the ring topology *lbest* PSO niching algorithms, one important parameter that needs to be specified is population size. Given a reasonably large population size, these PSOs are able to locate global optima (and/or local optima) reliably, especially for low dimensional problems. Fig. 22 shows that on f_{11} the inverted Shubert 2-D, with a population size of 450 or above, the ring topology *lbest* 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 [13], 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 peaks.

It is worth noting that the local hill-climber variants $r2pso-lhc$ and $r3pso-lhc$ performed better than $r2pso$ and $r3pso$ on f_{11} 2-D. This suggests that when handling low dimensional problems with multiple global optima in the presence of many local optima, it may be more effective to have multiple local hill climbers each optimizing independently than a niching algorithm with a more connected neighborhood topology.

VIII. Conclusion

Niching as an important technique for multimodal optimization has been used widely in the evolutionary computation research community. Many niching methods, however, are difficult to use in practice because they require prior knowledge to specify certain niching parameters. This paper has addressed this issue by proposing *lbest* PSO niching algorithms using a ring topology, which eliminate the need to specify any niching parameters. We have demonstrated that the *lbest* PSO algorithms with a ring topology are able to induce stable niching behavior. The *lbest* PSO algorithms with an *overlapping* ring topology (e.g., $r2pso$ and $r3pso$) are able to locate multiple global optima, given a reasonably large population size, whereas the *lbest* PSO algorithms with a *nonoverlapping* ring topology (e.g., $r2pso-lhc$ and $r3pso-lhc$) can be used to locate global as well as local optima, especially for low dimensional problems. Experimental studies carried out on a range of multimodal test functions suggest that the *lbest* PSO algorithms with a ring topology can provide comparable or better, and more consistent performance, than some existing niching PSO algorithms over these test functions. Even with a comparable or smaller population size, the proposed algorithms can outperform a niching algorithm using a fixed niche radius, in terms of success rate and the actual number of global optima found. More importantly, one major advantage over existing niching algorithms is that no niching parameters are required. This should pave the way for more widespread use of this kind of niching algorithms in real-world applications.

As far as we know, this paper is the first attempt showing that *lbest* PSOs with ring topology are able to induce stable niching behavior.[1] The findings of this research suggest that *local memory* and *slow communication topology* are the two key elements for the success of the proposed *lbest* PSO niching algorithms. In fact it does not have to be PSO functioning as a local optimizer. It is foreseeable that other population-based stochastic optimization methods characterized by these two key elements can be also used to induce stable niching behavior. Of course, the proposed ring topology based PSO is *no panacea*. If one has the domain knowledge on how to set niching parameters, using such knowledge might give even better performance. However, our study here assumes that there is no such prior knowledge readily available.

Since *lbest* PSO niching algorithms with an overlapping or nonoverlapping ring topology tend to generate multiple small

[1]The source code can be downloaded from the author's website: http://goanna.cs.rmit.edu.au/~xiaodong/rpso/.

niches, one interesting future research topic will be studying how to increase the search capability of small niches so that the performance of these niches will scale well with increasing dimensions. 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. It will be also interesting to apply the ring topology based PSO to tracking multiple peaks in a dynamic environment [16].

Acknowledgment

The author is grateful to Prof. K. Deb for his valuable comments and suggestions, via personal communication, and T. Nest for his careful proofreading of the paper.

References

[1] J. Kennedy and R. Eberhart, *Swarm Intelligence*. San Mateo, CA: Morgan Kaufmann, 2001.

[2] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Univ. Illinois, Urbana, IL, 1995. [Online]. Available: citeseer.ist.psu.edu/mahfoud95niching.html

[3] K. Koper and M. Wysession, "Multimodal function optimization with a niching genetic algorithm: a seis-mological example," *Bull. Seismol. Soc. Am.*, vol. 89, pp. 978–988, 1999.

[4] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proc. 1st IEEE Conf. Evol. Comput., IEEE World Congr. Comput. Intell.*, vol. 1. Piscataway, NJ, Jun. 1994, pp. 82–87. [Online]. Available: citeseer.ist.psu.edu/horn94niched.html

[5] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Univ. Michigan, Ann Arbor, MI, 1975.

[6] S. W. Mahfoud, "Crowding and preselection revisited," in *Proc. Parall. Prob. Solv. Nat. 2*, Amsterdam: North-Holland, 1992, pp. 27–36. [Online]. Available: citeseer.ist.psu.edu/mahfoud92crowding.html

[7] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. 2nd Int. Conf. Genet. Algorith.*, Cambridge, MA, 1987, pp. 41–49.

[8] D. Beasley, D. R. Bull, and R. R. Martin. (1993, Summer). A sequential niche technique for multimodal function optimization. *Evol. Comput.* [Online]. 1(2), pp. 101–125, 1993. Available: citeseer.ist.psu.edu/beasley93sequential.html

[9] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. 6th Int. Conf. Genet. Algorith.*, San Francisco, CA: Morgan Kaufmann, Jul. 1995, pp. 24–31. [Online]. Available: citeseer.ist.psu.edu/harik95finding.html

[10] M. Bessaou, A. Petrowski, and P. Siarry, "Island model cooperating with speciation for multimodal optimization," in *Proc. 6th Int. Conf. Parall. Prob. Solv. from Nat.: PPSN VI*, Paris, France: Springer Verlag, 2000, pp. 16–20. [Online]. Available: citeseer.ist.psu.edu/bessaou00island.html

[11] X. Yin and N. Germay, "A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization," in *Proc. Int. Conf. Artif. Neural Netwo. Genet. Algorith.*, 1993, pp. 450–457.

[12] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. 3rd IEEE Int. Conf. Evol. Comput.*, Nagoya, Japan, May 1996, pp. 798–803.

[13] J.-P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson, "A species conserving genetic algorithm for multimodal function optimization," *Evol. Comput.*, vol. 10, no. 3, pp. 207–234, 2002.

[14] A. E. R. Brits and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. 4th Asia-Pacif. Conf. Simul. Evol. Learn. (SEAL 2002)*, Singapore, Feb. 2002, pp. 692–696.

[15] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genet. Evol. Comput. Conf. 2004*, LNCS 3102. pp. 105–116.

[16] D. Parrott and X. Li, "Locating and tracking multiple dynamic optima by a particle swarm model using speciation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 440–458, Aug. 2006.

[17] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. IEEE Int. Conf. Evol. Comput.*, La Jolla, CA, 2000, pp. 303–308.

[18] R. Brits, A. Negelbrecht, and F. van den Bergh, "Locating multiple optima using particle swarm optimization," *Appl. Math. Comput.*, vol. 189, pp. 1859–1883, 2007.

[19] B. Sareni and L. Krahenbuhl, "Fitness sharing and niching methods revisited," *IEEE Trans. Evol. Comput.*, vol. 2, no. 3, pp. 97–106, Sep. 1998.

[20] A. Engelbrecht, *Fundamentals of Computational Swarm Intelligence*. New York: Wiley, 2005.

[21] M. Clerc, *Particle Swarm Optimization*. London, U.K.: ISTE Ltd., 2006.

[22] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromach. Hum. Sci.*, Nagoya, Japan, 1995, pp. 39–43.

[23] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. 2002 Cong. Evol. Comput.*, 2002, pp. 1671–1675.

[24] A. Engelbrecht, B. Masiye, and G. Pampara, "Niching ability of basic particle swarm optimization algorithms," in *Proc. IEEE Swarm Intell. Symp., 2005*, Pretoria, South Africa, Jun. 2005, pp. 1–4.

[25] D. Cavicchio, "Adapting search using simulated evolution," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1970.

[26] J. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press, 1975.

[27] D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," in *Proc. Parall. Prob. Solv. Nat. 2 (PPSN 2)*. Amsterdam: Elsevier Science Publishers, B. V., 1992. [Online]. Available: citeseer.ist.psu.edu/goldberg92massive.html

[28] P. J. Darwen and X. Yao, "A Dilemma for Fitness Sharing with a Scaling Function," in *Proc. 2nd IEEE Int. Conf. Evol. Comput.*, Piscataway, NJ, 1995. [Online]. Available: citeseer.ist.psu.edu/darwen95dilemma.html

[29] G. Singh and K. Deb, "Comparisons of multi-modal optimization algorithms based on evolutionary algorithms," in *Proc. Genet. Evol. Comput. Conf. 2006 (GECCO '06)*, Washington D.C., pp. 1305–1312.

[30] M. Jelasity and J. Dombi (1998). GAS, a concept on modeling species in genetic algorithms. *Artif. Intell.* [Online]. *99(1)*. Available: citeseer.ist.psu.edu/jelasity98gas.html

[31] O. Shir and T. Bäck, "Niche radius adaptation in the cms-es niching algorithm," in *Proc. 9th Int. Conf. Parall. Prob. Solv. Nat. (PPSN)*, LNCS 4193. Reykjavik, Iceland: Springer, 2006, pp. 142–151.

[32] S. Bird and X. Li, "Adaptively choosing niching parameters in a PSO," in *Proc. Genet. Evol. Comput. Conf. (GECCO '06)*, Seattle, WA: ACM, pp. 3–10. [Online]. Available: http://doi.acm.org/10.1145/1143997.1143999

[33] S. Bird and X. Li, "Enhancing the robustness of a speciation-based PSO," in *Proc. 2006 IEEE Cong. Evol. Comput.*, Vancouver, BC, Canada: IEEE Press, Jul. 16–21, 2006, pp. 843–850. [Online]. Available: http://ieeexplore.ieee.org/servlet/opac?punumber=11108

[34] M. Clerc and J. Kennedy, "The particle swarm—explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[35] K. Parsopoulos and M. Vrahatis, "Modification of the particle swarm optimizer for locating all the global minima," *Artificial Neural Networks and Genetic Algorithms*, Springer, 2001, pp. 324–327.

[36] K. Parsopoulos and M. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.

[37] R. Brits, A. Negelbrecht, and F. van den Bergh, "Solving systems of unconstrained equations using particle swarm optimizers," in *Proc. IEEE Conf. Syst., Man, Cybernet.*, Pretoria, South Africa, Oct. 2002, pp. 102–107.

[38] X. Li, "Multimodal function optimization based on fitness-euclidean distance ratio," in *Proc. Genet. Evol. Comput. Conf. 2007*, pp. 78–85.

[39] I. Schoeman and A. Engelbrecht, "Using vector operations to identify niches for particle swarm optimization," in *Proc. 2004 IEEE Conf. Cybernet. Intell. Syst.*, Singapore, Dec. 2004, pp. 361–366.

[40] Y. Davidor, "A naturally occurring niche & species phenomenon: The model and first results," in *Proc. 4th Int. Conf. Genet. Algorith.*, San Mateo, CA: Morgan Kaufmann, Jul. 1991, pp. 257–263.

[41] Z. Perry, "Experimental study of speciation in ecological niche theory using genetic algorithms (doctoral dissertation)," Ph.D. dissertation, Univ. Michigan, Ann Arbor, 1984.

[42] K. Deb, "Genetic algorithms in multimodal function optimization, the Clearinghouse for Genetic Algorithms," M.S thesis and Rep. 89002, Univ. Alabama, Tuscaloosa, 1989.

[43] K. Deb and D. Goldberg, "An investigation of niche and species formation in genetic function optimization," in *Proc. 3rd Int. Conf. Genet. Algorith.*, 1989, pp. 42–50.

[44] A. W. Iorio and X. Li, "Rotated test problems for assessing the performance of multiobjective optimization algorithms," in *Proc. 8th Annu. Conf. Genet. Evol. Comput. (GECCO '06)*, New York, pp. 683–690.

[45] D. Ackley, "An empirical study of bit vector function optimization, in *Genetic Algorithms Simulated Annealing*, London, U.K.: Pitman, 1987, pp. 170–204.

[46] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.

[47] A. Törn and A. Zilinskas, *Global Optimization, volume 350*. New York: Springer-Verlag, 1987.

[48] S. Jason and M. Middendorf, "On trajectories of particles in PSO," in *Proc. 2007 IEEE Swarm Intell. Symp. (SIS 2007)*. Piscataway, NJ: IEEE Service Center, Feb. 2007, pp. 38–44.

Xiaodong Li (SM'07) received the B.Sc. degree from Xidian University, Xi'an, China, in 1988, and the Dip.Com. and Ph.D. degrees in information science from the University of Otago, Dunedin, New Zealand, in 1992 and 1998, respectively.

Currently, he is with the School of Computer Science and Information Technology, RMIT University, Melbourne, Australia. His research interests include evolutionary computation, neural networks, complex systems, and swarm intelligence.

Dr. Li is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *International Journal of Swarm Intelligence Research*. He is a Member of the IASR Board of Editors for the Journal of Advanced Research in Evolutionary Algorithms.