A Dynamic Archive Based Niching Particle Swarm Optimizer Using a Small Population Size

Zhaolin Zhai **Xiaodong Li**

School of Computer Science and Information Technology
RMIT University, Melbourne, VIC 3001, Australia
Email: zhaolin.zhai@student.rmit.edu.au; xiaodong.li@rmit.edu.au

Abstract

Many niching techniques have been proposed to solve multimodal optimization problems in the evolutionary computing community. However, these niching methods often depend on large population sizes to locate many more optima. This paper presents a particle swarm optimizer (PSO) niching algorithm only using a dynamic archive, without relying on a large population size to locate numerous optima. To do this, we record found optima in the dynamic archive, and allow particles in converged sub-swarms to be re-randomized to explore undiscovered parts of the search space during a run. This algorithm is compared with $lbest$ PSOs with a ring topology (LPRT). Empirical results indicate that the proposed niching algorithm outperforms LPRT on several benchmark multimodal functions with large numbers of optima, when using a small population size.

Keywords: Particle Swarm Optimization, Multimodal Optimization, Evolutionary Computation, a Dynamic Archive.

1 Introduction

Optimization techniques, such as Evolutionary Algorithms (EAs) (Back et al. 1997), Particle swarm optimization (PSO) (Kennedy & Eberhart 1995), and Differential Evolution (Price et al. 2005)have proven to be successful in solving difficult global optimzation problems, typically chacterized by searching a single global optimum. However, many real optimization problems in science and engineering do have more than one global optimal solution known as multimodal problems. For instance, in the field of wing design in aviation industry, there may be several different solutions which could perform equally well. It is desirable to locate as many optimal solutions as possible, so engineers are able to select the best solution depending on the preferred design variable ranges. To address this issue, a number of approaches to find multiple solutions have been proposed (Li et al. 2002, R. Brits & van den Bergh 2002, Bird & Li 2006, Li 2010). These approaches are generally referred to as niching or speciation algorithms. The notion of niching or speciation is originated from ecological science, in which all kinds of species (a class of individuals with common characteristics (Li et al. 2002)) are considered to be evolved in parallel by competition and species form different niches over time. In the context of EC, niches refer subpopulations formed around global or local optima, and a niching algorithm aims to identify and maintain equally good solutions stably throughout a run (Mahfoud 1995). In a typical EC based niching algorithm, individuals in a population are partitioned someway, generating a set of subpopulation known as species. These species are expected to be converged around different global or local optima (attraction basins) in the search space for locating different potential solutions.

Currently, existing niching algorithms largely depend on using a large population size to find large numbers of optima in the search space. However, without the prior knowledge on numbers of optimal solutions to the problem, it is extremely hard to determine how many individuals or particles are sufficient for a multimodal problem. In particular, for problems with numerous global optima, it is difficult for existing niching algorithms to locate all optima, if the population size is not sufficiently large. Furthermore, most reported experimental results on niching algorithms, is limited to low dimensional problems or higher dimensional problems with a small number of global optima (Bird & Li 2006, Li et al. 2002, Schoeman & Engelbrecht 2005, Li 2010). However, in many cases, as the dimensionality increases, the number of global and local optima in multimodal problems may go up quickly. For instance, the inverted Vincent function has 6^n number of global optima, where n is the number of dimensions.

This paper introduces a dynamic archive based PSO niching algorithm ($rpso - sp$) to alleviate the population dependence problem in niching algorithms. $rpso - sp$ is able to handle multimodal optimization problems using a dynamic archive for saving best found solutions. To make each particle keep doing search in a run, a multi-start technique is employed in $rpso - sp$ through re-randomizing converged sub-population, allowing sub-population explore undiscovered parts of the search space continuously. The paper is organized as follows. Section 2 gives an introduction to the basic PSO algorithms. Section 3 provides a review of some state-of-the art PSO niching techniques. Section 4 describes the newly proposed $rpso-sp$ niching algorithm. Experimental results and analysis are provided in Section 5 and 6, followed by conclusions in Section 7.

2 Particle Swarm Optimization

Particle swarm optimization (PSO) is a recently developed stochastic optimization algorithm first introduced by Kennedy & Eberhart (1995), inspired by social behaviors observed among insects and animals such as bird flocking or fish schooling. Like Genetic Algorithms (GAs), PSO uses a population of agents, referred to as particles, to form a swarm. Each particle in the swarm represents a candidate solution to an optimization problem. In a standard PSO algorithm, each particle moves through the search space by adjusting its position based on its own experience that of neighborhood particles. We denote the

i^{th} particle in the swarm as $\vec{\mathbf{x}}_i$, which is evaluated as a potential solution at each iteration. The i^{th} particle moves with a velocity $\vec{\mathbf{v}}_i$ toward promising regions by consulting the best position $\vec{\mathbf{p}}_i$ (the position giving the best fitness value so far) found by itself and that of its neighbors $\vec{\mathbf{p}}_g$. As to deciding neighbors for each particle, two types of neighborhood topologies are extensively used in PSO. In a global best ($gbest$) PSO, each particle influences every other particle; $\vec{\mathbf{p}}_g$ is obtained from the entire swarm. In the local best ($lbest$) PSO, each particle influences its immediate or close neighbors. We adopted the constriction coefficient variation of PSO (Clerc & Kennedy 2002) in this paper. Updating $\vec{\mathbf{v}}_i$ and $\vec{\mathbf{x}}_i$ for the i^{th} particle is based on the following equations:

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

$$\vec{\mathbf{x}}_i = \vec{\mathbf{x}}_i + \vec{\mathbf{v}}_i \quad (2)$$

where $\vec{\mathbf{R}}_1[0,\varphi_1]$ and $\vec{\mathbf{R}}_2[0,\varphi_2]$ generate two vectors containing random values uniformly distributed in the range [0, φ_1] and [0, φ_2] respectively. φ_1 and φ_2 are usually set to $\frac{\varphi}{2}$. φ is a positive constant. $\otimes$ indicates point-wise vector multiplication. A constriction coefficient χ is used to restrict particles's visiting regions (commonly $\chi = 0.7298$). χ is calculated according to $\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$, where $\varphi = \varphi_1 + \varphi_2 = 4.1$ (Clerc & Kennedy 2002).

3 PSO Niching Techniques

PSO niching methods commonly attempt to locate multiple optima to multimodal problems by partitioning a swarm into a number of subs-swarms. These sub-swarms run independently as local optimizers to find multiple optimal solutions in parallel. Currently, a variety of niching PSO algorithms have been developed. Parsopoulos and Vrahitis introduced a method to identify particles as potential solutions when the fitness value of this solution is lower than a predefined threshold value (Parsopoulos & Vrahatis 2001). Brits et al proposed NichePSO (R. Brits & van den Bergh 2002) and they extended Parsopoulos and Vrahitis' niching method by adopting multiple subswarms generated from a main swarm. A sub-swarm is created around a particle with little change of its fitness over a number of iterations. Then a set of produced subswarms do local search in parallel. Another niching PSO algorithm is species particle swarm optimizer (SPSO) proposed by Li (Li 2004). The basic idea is similar to a species conserving genetic algorithm (SCGA) in (Li et al. 2002), except that PSO is used as a local optimizer instead of using GA. In SPSO, a niche radius must be predefined. It is a typical niching parameter, used to specify the upper bound on the distance between two individuals being considered to be in the same niche (Li et al. 2002). To avoid predefining such a parameter, Li introduced an $lbest$ PSO (as described in Section 2) with a ring topology (LPRT), adopting a ring topology to form niches without requiring a niche radius (Li 2010).

4 A Dynamic Archive based PSO Niching Algorithm ($rpso-sp$)

The reason existing PSO niching techniques often requiring a large population size (Li et al. 2002, Bird & Li 2006, Li 2004, 2010), for solving complex multimodal problems is that a large particle swarm is more likely to generate a great number of sub-swarms. The more sub-swarms a niching method produces, the more local or global optima the niching method possibly can find. However, without the prior knowledge on numbers of optimal solutions to the problem, it is almost impossible to figure out the appropriate number of particles for a niching algorithm. To alleviate such a problem, a new niching PSO algorithm, $rpso-sp$, is proposed in this paper.

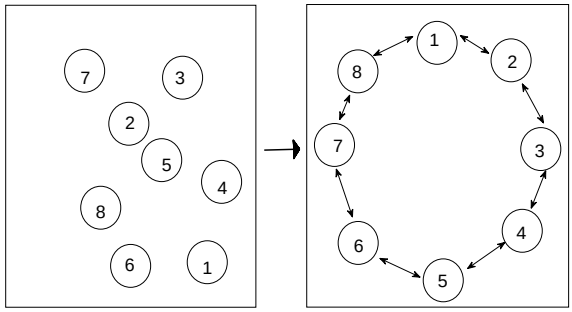

Figure 1: An example of mapping 8 particles in the swarm to a ring topology. Assuming two dimensional search space is used

4.1 Constructing sub-swarms by using a ring topology

$rpso-sp$ adopts LPRT's idea on forming sub-swarms for two reasons: simplicity and niching parameter-free (Li 2010). Existing niching algorithms commonly use a relatively complex procedure to dynamically group particles as sub-swarms (Bird & Li 2006, Schoeman & Engelbrecht 2005) based on given niching parameters and particles' current positions. Practically, many niching algorithms group particles within a threshold distance (known as a niche radius) in the search space as sub-swarms. LPRT instead maps all particles in the search space to a ring topology and Figure 1 gives such an example. Left side of the figure shows particles' actual positions in the search space (assuming two dimensional search space); the right side shows how a population of 8 particles is mapped into a ring topology. Then LPRT forms sub-swarms by grouping a fixed number of particles, which have the closest index values on a ring topology. Figure 2 and Figure 3 illustrate two variants instances of LPRT: $r3pso$ and $r3pso-lhc$ (Li 2010). As shown in these two Figures, both $r3pso$ and $r3pso-lhc$ have sub-swarms consisting of 3 particles except that the number of particles on the tail of a ring is less than 3. For instance, sub-swarm C only contains 2 particles, shown in Figure 3. As to the difference between these two variants, $r3pso$ has overlapped sub-swarms which are formed by any particle and its left and right neighbors on the ring topology indicated in Figure 2. $r3pso-lhc$ is the same as $r3pso$, but without overlapping neighbors shown in Figure 3. Each particle in $r3pso-lhc$ only belongs to one sub-swarm. Multiple formed sub-swarms in $r3pso-lhc$ do local search independently, like local hill climbers (lhc). For detailed description about LPRT, please refer to (Li 2010). LPRT's method uncovers two new features, compared with other niching algorithms. Firstly, members in each sub-swarm remain unchanged throughout a run. Secondly, particles in other parts of the search space have chances to form a sub-swarm, as LPRT creates sub-swarms regarding particles' indexes on a ring topology instead of particles' actual positions in the search space. This feature is clearly shown in Figure 1.

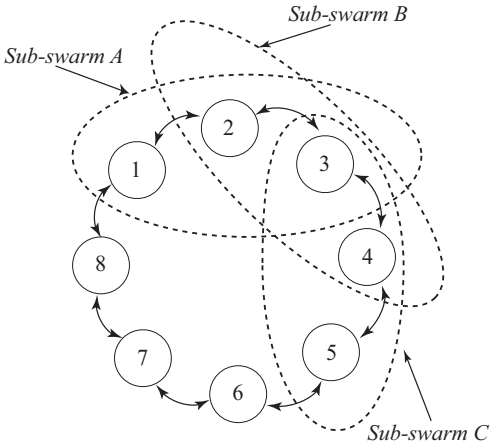

Figure 2: An example of $r3pso$. Sub-swarms A, B and C consist of 3 particles and they are overlapped.

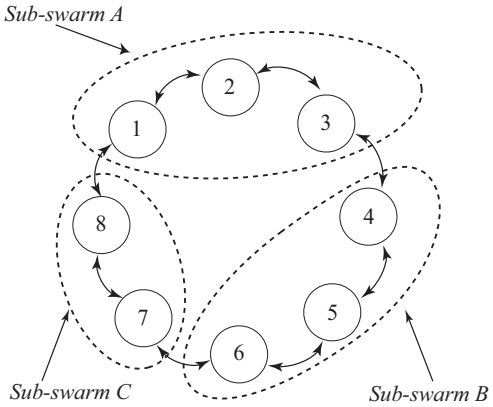

Figure 3: An example of $r3pso - lhc$. It is the same as $r3pso$, but without overlapping neighbors. All sub-swarms A, B and C do local search independently, like local hill climbers (lhc).

4.2 A Dynamic Archive Recording Found Optima

$rpso - sp$ extends LPRT by utilizing a dynamic archive (*DA*) to record optima found by converged sub-swarms. And these converged sub-swarms are re-randomized to search other parts of the search space, achieved through a multi-start technique. As far as the basic concept of niching is concerned, niching algorithms can be considered to be either sequential or parallel (Brits et al. 2007). If a PSO niching method repetitively applies $gbest$ PSO to the search space until all global optima are found, this algorithm is categorized as being sequential niching. In contrast, if a PSO niching method partitions the swarm into several sub-swarms for forming different niches simultaneously, this method is in the domain of parallel niching. Obviously, $rpso-sp$ presented in this paper holds characteristics from both sequential niching (employing a multi-start technique) and parallel niching (using a ring topology to form sub-swarms). So $rpso - sp$ is a hybrid of sequential and parallel niching ideas. In addition, existing parallel niching algorithms commonly record found optimal solutions in members of the population. This could possibly lead to identified solutions missing due to operations in PSO. To better maintain equally good solutions, a dynamic archive is used in $rpso - sp$ for preservation of optimal solutions. By doing so, $rpso - sp$ can lower the risk of losing found optimal solutions.

Additionally, to make the comparison between $rpso - sp$ and $LPRT$ clear in Section 6, variants of $rpso - sp$ follow the similar naming rules as $LPRT$ does (Li 2010). It is noticed that numbers in variants names indicates the number of members in each sub-swarm. Like $LPRT$, $rpso-sp$ also has several variants with varying sub-swarm sizes. $r3pso - sp$ and $r3pso - sp - lhc$ are two typical variants of $rpso - sp$. They are the same as $r3pso$ and $r3pso - lhc$ respectively, except that a dynamic archive is used in each variant for recording optima found by converged sub-swarms.

Our proposed $rpso - sp$ algorithm works generally in the following steps:

Step 1: Initialize particles to random positions in the search space and use a ring topology to form sub-swarms. At the same time, an empty dynamic archive *DA* is built.

Step 2: Sub-swarms run $gbest$ PSOs in parallel. The best solution found by the converged sub-swarm is checked with the *DA* to confirm that whether a found solution is qualified to add into the *DA*. Then converged sub-swarms are re-randomized to do search again. In terms of identifying converged sub-swarms, a sub-swarm is considered to be converged when the velocity of any particle in the sub-swarm approximately reduces to 0. The reason behind this is simple. Assume that a particle in a sub-swarm stops moving in the search space, this implies that no better positions can be found either by other particles in the sub-swarm or by the particle itself, according to Equation (1). Furthermore, the stopped particle almost is a leading particle with the best solution in the sub-swarm.

Step 3: When stopping criteria are met, $rpso-sp$ terminates.

The detailed algorithm for building a *DA* is presented in Algorithm 1; it is performed at each iteration step in $rpso - sp$. We assume maximization in this paper.

As can be seen in Algorithm 1, a dynamic archive S records two types of solutions p found by sub-swarms.

Type 1: p has a better fitness value than a threshold δ ($f(p) > \delta$). δ holds the best fitness value found by sub-swarms so far. Under this condition, p is the best solution found since a run starts. p will replace the saved solution s in the dynamic archive if p and s are considered to be similar solutions. Namely Euclidean distance between p and s is smaller than identification radius R ($||p - s|| \leq R$; see equations (3)–(5)). If it is not the case, it means p has no similar solution in S. So p will be simply added into S. R is used to identify different solutions.

Type 2: p has a relatively good fitness value compared with recorded solutions in the dynamic archive $|f(p) - \delta| < \epsilon$. ϵ is set by users as acceptable accuracy, used to identify a qualified solution p. In this case, if p and s are similar $||p - s|| \leq R$, and p has better fitness value ($f(p) > f(s)$), then p will replace s in the dynamic archive. If p has no similar solution in S, p will be directly added into S

Equation (3) calculates the minimum distance $dist_i$ between each particle k_i and other particles k_j. K represents a set, consisting of n particles in the swarm. $||k_i - k_j||$ computes Euclidean distance between k_i and k_j in the search space. Then the average of minimum distances r_m in the m^{th} iteration of a run is obtained over Equation (3) and Equation (4), suggested by Bird & Li (2006). r_m in fact reflects the convergence degree of all particles in the swarm. We assume that the most of sub-swarms have

```
input  : p - a potential solution obtained from converged sub-swarms
output : S - a list of found solutions in a dynamic archive
begin
    found ← FALSE;   update ← FALSE;
    if S = ∅ then
        S ← S ∪ {p};   δ ← f(p);
    end
    else
        if f(p) > δ then
            δ ← f(p);   update ← TRUE;
        end
        if update or |f(p) − δ| < ε then
            for each s ∈ S do
                if ||p − s|| ≤ R then
                    if f(p) > f(s) then
                        s ← p;
                        found ← TRUE;
                        break;
                    end
                    else
                        found ← TRUE;
                        break;
                    end
                end
            end
            if not found then
                S ← S ∪ {p};
            end
        end
    end
end
```

Algorithm 1: Pseudocode for building a dynamic archive.

already converged on a number of niches. Then the minimum distance between each particle and other particles, is close to zero. Accordingly, the average of minimum distances r_m is reaching zero. In short, while r_m is approaching zero, the convergence degree becomes very high.

$$dist_i = min\{||k_i - k_j||; \forall k_i, k_j \in K \wedge k_i \neq k_j\} \quad (3)$$

$$r_m = \frac{\sum_{i=1}^{n} dist_i}{n} \quad (4)$$

$$R = min\{r_1, r_2, \ldots r_m\} \quad (5)$$

The identification radius R employed in Algorithm 1, is used for deciding that whether a newly found solution is already in the *DA*. R denotes the smallest value among the average of minimum distances produced up to the m^{th} iterations of a run, shown in Equation (5). The reason why we choose R in this way is easily explained. In the first half of a run, it is highly possible that niches on the big attraction basins in multimodal problems have been identified, as big attraction basins easily absorb more particles than small attraction basins. Along with more and more niches having been found, discovering niches located on the smaller attraction basins becomes a major challenge. To handle this problem, a smaller and smaller identification radius actually is more preferrable. Therefore, it makes sense to choose R in this way which can be adaptively chosen in a run.

5 Experimental setup

To measure the performance of PSO niching algorithms, a set of extensively used multimodal benchmark functions was used. Table 1 indicates the test functions used in this study ranging from simple to more challenging ones. f_1 has 5 evenly spaced global peaks. f_2 has 5 unevenly spaced global peaks. f_3 has four unevenly spaced global peaks without any local peaks. The inverted Shubert function f_4 and the inverted Vincent function f_5 are more challenging N-dimensional multimodal functions. Due to having a large number of unevenly spaced local and global optima when the function dimensionality n increases, both f_4 and f_5 are widely used in this study. For inverted Shubert 2D function, it has 18 global peaks spaced in 9 groups. The distance between global peaks in the same group is much closer than global peaks in the different groups. In general, f_4 has $n \cdot 3^n$ global peaks in 3^n groups. f_5 has 6^n unevenly spaced global peaks. f_6 only keeps a single global peak when its dimension increases. f_7 has 5^n clearly known global peaks evenly spaced in n-dimensional search space without local peaks.

We used *peak ratio* (PR) as the performance measurement throughout experiments in this study. Peak ratio defines the proportion of found peaks to the total number of known peaks. For hard test functions with a large number of global peaks, peak ratio is more informative when comparing the niching ability between different niching algorithms. To count the number of found peaks for each algorithm, an acceptance threshold γ was set to identify whether a found solution is close enough to a known optimum. If the fitness of a found solution is within γ of the desired global best fitness, this solution is considered found.

6 Experimental Results

Five sets of experiments were carried out. Firstly, We evaluated several $rpso - sp$ variants and chose a relatively good performer as the representative of $rpso - sp$. We then compared $rpso - sp$ and $LPRT$ in four aspects: basic niching ability, sensitivity to population size, niching ability by using a small population size and scalability on high dimensional multimodal functions.

In this study, all experimental results are average values over 50 runs. MNFE and NFE indicate the maximum number of function evaluations to be allowed in a run and number of function evaluations spent during a run respectively. PS and PR denote population size and peak ratio respectively.

6.1 Choosing a $rpso - sp$ Variant comparing with $LPRT$

To make an informed decision on which variant of $rpso - sp$ should be adopted, a set of $rpso - sp$ variants were tested on the inverted Shubert 3D function $f_4(3D)$ and a relatively better performer was chosen as the base algorithm for $rpso - sp$. As can be seen in Figure 4, $r3pso - sp$ (like $r3pso$), performs much worse than the variant $r3pso - sp - lhc$ no matter how big sub-swarm sizes are. It is clear that $r3pso - sp - lhc$ performs reasonably well although it is not the best one among $rpso - sp - lhc$ variants. $r3pso - sp - lhc$ thereby is chosen as the core niching algorithm used in our experiments and it is simply denoted as $rpso - sp$. It is noticed that the performance of variants does not simply increase with increasing sizes of sub-swarms. For example, $r20pso - sp - lhc$ performs much worse than $r3pso - sp - lhc$ does. In this paper, we focus on comparison between $rpso - sp$ and two variants of $LPRT$:

Table 1: Test functions. n and GP are function dimensions and the number of global peaks respectively.

Name	Function	Range	GP
Equal Maxima (Deb 1989)	$f_1(x) = \sin^6(5\pi x)$	$[0, 1]$	5
Uneven Maxima (Deb 1989)	$f_2(x) = sin^6(5\pi(x^{3/4} - 0.05))$.	$[0, 1]$	5
Himmelblau's function (Deb 1989)	$f_3(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$	$[-6, 6]$	4
Inverted Shubert function (Li et al. 2002)	$f_4(\vec{x}) = -\prod_{i=1}^{n}\sum_{j=1}^{5} j\cos[(j+1)x_i + j]$	$[-10, 10]$	$n \cdot 3^n$
Inverted Vincent function (Li 2010)	$f_5(\vec{x}) = \frac{1}{n}\sum_{i=1}^{n}\sin(10 \cdot \log(x_i))$	$[0.25, 10]$	6^n
Inverted Rastrigin function (Li 2010)	$f_6(\vec{x}) = -\sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	$[-1.5, 1.5]$	1
Inverted Deb's 1st function (Deb 1989)	$f_7(\vec{x}) = 1 - \frac{1}{n}\sum_{i=1}^{n}(1 - \sin^6(5\pi x_i))$	$[0, 1]$	5^n

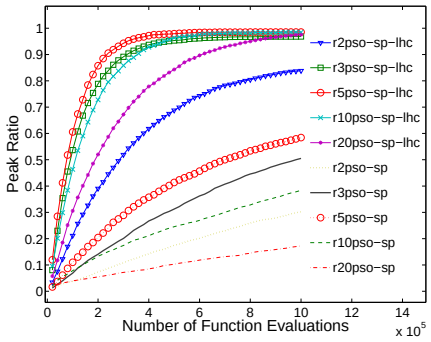

Figure 4: The performance of $rpso - sp$'s variants on the inverted Shubert 3D function. $\gamma = 0.1$, $PS = 100$, $MNFE = 1,000,000$.

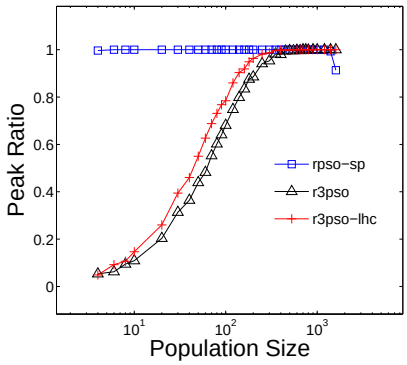

Figure 5: Peak ratios for Inverted Shubert 2D Function as PS increases from 4 to 1600. $\gamma = 0.0001$, $MNFE = 200,000$

$r3pso$ and $r3pso - lhc$, which showed the best performances among all $LPRT$ variants on a range of multimodal test functions (Li 2010).

6.2 Results on One or Two Dimensional Test Functions.

Table 2 shows that $r3pso$ and $r3pso - lhc$ are able to successfully locate all global peaks on $f_1(1D)$, $f_2(1D)$ and $f_6(2D)$ within $MNFE = 200,000$, as well as $rpso - sp$. However, they perform poorly on $f_4(2D)$ and $f_7(2D)$. Peak ratios for $r3pso$ and $r3pso - lhc$ on $f_7(2D)$ are only 46% and 49% respectively. Meanwhile, both $r3pso$ and $r3pso - lhc$ fail to find all global peaks on $f_3(2D)$ in several runs. In contrast, $rpso - sp$ performs consistently well on these nine benchmark functions and it can locate all global peaks within $MNFE = 200,000$.

6.3 The Effect of varying Population Sizes

To explore the effect of population size on the performance of niching algorithms, population sizes ranging from 4 to 1600 were used for $rpso - sp$, $r3pso$ and $r3pso - lhc$, on optimizing $f_4(2D)$ and $f_5(2D)$. Figure 5 and Figure 6 show $r3pso$ and $r3pso - lhc$ have a good niching ability only when population size is very large($PS > 250$ on $f_4(2D)$ and $PS > 1200$ on $f_5(2D)$). When smaller population sizes are used, PR drops dramatically for both variants. For example, when $PS < 10$, $PR < 0.2$ is for both variants on $f_4(2D)$. In contrast, $rpso - sp$ performs consistently well for different population sizes. Even when $PS < 10$, $rpso - sp$ is still able to find almost 18 optima on $f_4(2D)$.

To sum up, $rpso - sp$ shows a much lower sensitivity to population size than both $r3pso$ and $r3pso - lhc$. $rpso - sp$ performs consistently well when varying the population size. On the other hand, the performance of $r3pso$ and $r3pso - lhc$ are largely dependent on population sizes. Without a large enough PS, these algorithms may fail to achieve desired results.

6.4 The effect of increasing number of evaluations

Since the goal of a niching algorithm is to find and maintain niches on all global optima, we designed a set of experiments to monitor peak ratios in the whole run for each algorithm by using a relatively small population size: $PS = 50$. Figure 7 and Figure 8 demonstrate the niching behaviors for each algorithm. Although $rpso - sp$ is unable to find all global optima on $f_4(3D)$ and $f_5(3D)$ in most cases, it performs far better than $r3pso$ and $r3pso - lhc$. Furthermore, Figure 8 displays that $rpso - sp$ has a potential to obtain higher peak ratios if more evaluations are given.

As expected, one of advantages $rpso - sp$ has is that it is able to remember all found optima. In other words, the issue of maintaining niches in niching algorithms is handled by employing a dynamic archive to save found optima. More importantly, a population size is no longer a crucial parameter for $rpso - sp$. Since it is able to find more optima continuously, without depending on the specified population size. Finally, compared with $r3pso - lhc$ and $r3pso$, $rpso - sp$ can reach higher PR within a small number of NFE as shown in both Figure 7 and Figure 8.

6.5 The effect of increasing dimensionalities

To examine the scalability of niching PSO variants, a set of experiments were conducted on f_4 and f_5 by increasing the dimension from 2 to 5, as shown in Figure 9 and Figure 10. $rpso - sp$ still performs better than other niching variants, although the performance of all three variants

Table 2: Peak ratios on low dimensional functions for $r3pso$, $r3pso - lhc$ and $rpso - sp$. $\gamma = 0.0001$, $PS = 50$, $MNFE = 200,000$.

	$f_1(1D)$	$f_2(1D)$	$f_3(2D)$	$f_4(2D)$	$f_5(1D)$	$f_6(2D)$	$f_7(2D)$
$r3pso$	100%	100%	85%	51%	78%	100%	*46%*
$r3pso - lhc$	100%	100%	98%	54%	90%	100%	*49%*
$rpso - sp$	100%	100%	100%	100%	100%	100%	100%

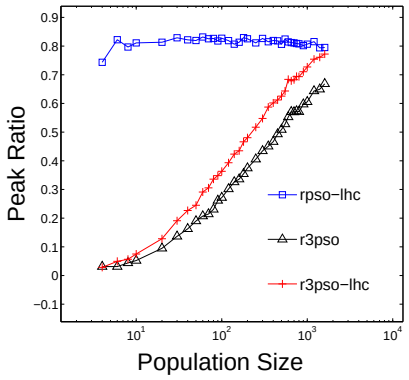

Figure 6: Peak ratios for Inverted Vincent 2D Function as PS increases from 4 to 1600. $\gamma = 0.0001$, $MNFE = 500,000$

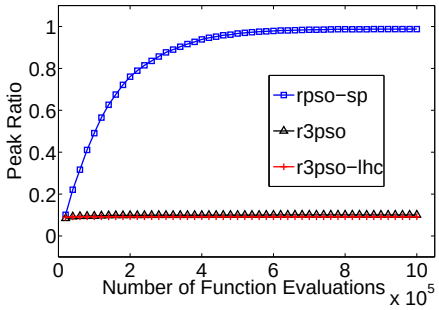

Figure 7: Peak ratios for Inverted Shubert 3D as the number of function evaluations increases. $MNFE = 1,000,000$, $PS = 50$, $\gamma = 0.01$.

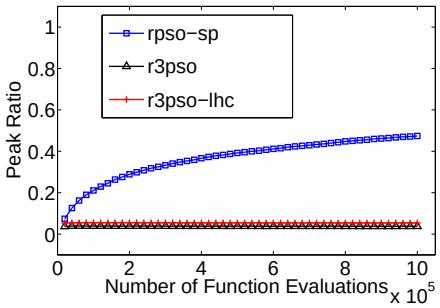

Figure 8: Peak ratios for Inverted Vincent 3D as the number of function evaluations increases. $MNFE = 1,000,000$, $PS = 50$, $\gamma = 0.0001$.

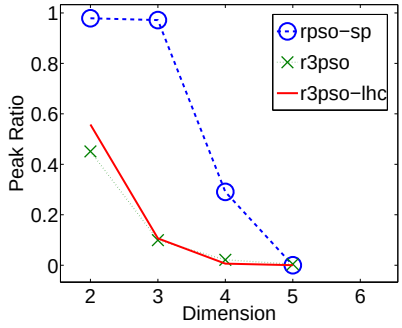

Figure 9: Peak ratios for Inverted Shubert Function (f_4) as the dimensionality increases from 2 to 5. $\gamma = 0.1$, $MNFE = 1,000,000$, $PS = 50$.

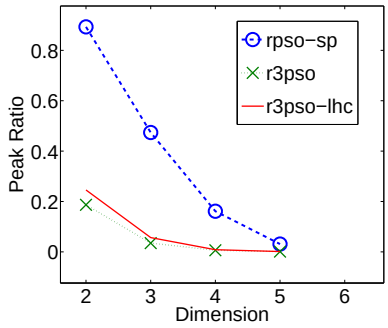

Figure 10: Peak ratios for Inverted Vincent Function (f_5) as the dimensionality increases from 2 to 5. $\gamma = 0.0001$, $MNFE = 1,000,000$, $PS = 50$.

degrade quickly with increasing dimensions. It is noticeable that $rpso - sp$ managed to keep a peak ratio ≈ 0.2 on both $f_4(4D)$ and $f_5(4D)$. Accordingly, the absolute number of optima found is around 94 and 210 on $f_4(4D)$ and $f_5(4D)$ respectively. This implies that $rpso - sp$ still maintain a relatively better scalability to higher dimensional problems even when the population size is much smaller than the number of known global peaks on benchmark functions. However, all PSO niching variants reach much lower peak ratios when $dimension = 5$.

7 Conclusions

The primary advantage of $rpso - sp$ over previous niching techniques is that it does not depend on a large population size to optimize multimodal functions with a large number of global optima. Our experiments clearly demonstrate that $rpso - sp$ is able to provide competitive performance even when only small population sizes have been used. Although the performance of $rpso - sp$ drops quickly as the function dimensionality increases, it per-

forms consistently well or better than the ring topology based PSO niching algorithms without using a dynamic archive. Since a user does not have to specify niching parameters and only a small population size is required, $rpso - sp$ holds a great promise in real world problem solving. Future work will investigate how to further enhance the scalability of $rpso - sp$ and apply $rpso - sp$ to real–world problems solving.

References

Back, T., Fogel, D. B. & Michalewicz, Z., eds (1997), *Handbook of Evolutionary Computation*, IOP Publishing Ltd., Bristol, UK, UK.

Bird, S. & Li, X. (2006), Adaptively choosing niching parameters in a PSO, *in* M. Cattolico, ed., 'Genetic and Evolutionary Computation Conference, GECCO 2006, Proceedings, Seattle, Washington, USA, July 8-12, 2006', ACM, pp. 3–10.

Brits, R., Engelbrecht, A. & van den Bergh, F. (2007), 'Locating multiple optima using particle swarm optimization', *Applied Mathematics and Computation* **189**(2), 1859 – 1883.

Clerc, M. & Kennedy, J. (2002), 'The particle swarm - explosion, stability, and convergence in a multidimensional complex space', *IEEE Trans. on Evol. Comput.* **6**, 58–73.

Deb, K. (1989), Genetic Algorithms in multimodal function optimization (Master thesis and TCGA Report No. 89002), PhD thesis, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms.

Kennedy, J. & Eberhart, R. (1995), Particle swarm optimization, *in* 'Proc. Conf. IEEE Int Neural Networks', Vol. 4, pp. 1942–1948.

Li, J.-P., Balazs, M. E., Parks, G. T. & Clarkson, P. J. (2002), 'A species conserving genetic algorithm for multimodal function optimization', *Evol. Comput.* **10**(3), 207–234.

Li, X. (2004), Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, *in* K. Deb, ed., 'Proc. of Genetic and Evolutionary Computation Conference 2004(LNCS 3102)', pp. 105–116.

Li, X. (2010), 'Niching without niching parameters: Particle swarm optimization using a ring topology', *Evolutionary Computation, IEEE Transactions on* **14**(1), 150 –169.

Mahfoud, S. W. (1995), Niching methods for genetic algorithms, PhD thesis, Urbana, IL, USA.

Parsopoulos, K. & Vrahatis, M. (2001), Modification of the particle swarm optimizer for locating all the global minima, *in* R. N. M. K. V. Kurkova, N. Steele, ed., 'Artificial Neural Networks and Genetic Algorithms', Springer, pp. 324–327.

Price, K., Storn, R. M. & Lampinen, J. A. (2005), *Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA.

R. Brits, A. E. & van den Bergh, F. (2002), A niching particle swarm optimizer, *in* 'Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002(SEAL 2002)', pp. 692–696.

Schoeman, I. & Engelbrecht, A. (2005), 'A parallel vector-based particle swarm optimizer', pp. 268–271.