Comparing $lbest$ PSO Niching algorithms Using Different Position Update Rules

Xiaodong Li and Kalyanmoy Deb

Abstract— **Niching is an important technique for multimodal optimization in Evolutionary Computation. Most existing niching algorithms are evaluated using only 1 or 2 dimensional multimodal functions. However, it remains unclear how these niching algorithms perform on higher dimensional multimodal problems. This paper compares several schemes of PSO update rules, and examines the effects of incorporating these schemes into a $lbest$ PSO niching algorithm using a ring topology. Subsequently a new Cauchy and Gaussian distributions based PSO (CGPSO) is proposed. Our experiments suggest that CGPSO seems to be able to locate more global peaks than other PSO variants on multimodal functions which typically have many global peaks but very few local peaks.**

I. Introduction

Most real-world problems are *multimodal* by nature, i.e., multiple equally good solutions (global or local) exist. It may be desirable for a decision maker (DM) to locate all global solutions or satisfactory solutions so that the DM can then choose the suitable one from the pool of multiple solutions depending on the preferred decision variable ranges in the decision space.

Numerous techniques have been developed in the past for locating multiple optima. Two most well-known such techniques are probably *crowding* [1] and *fitness sharing* [2]. These techniques are commonly referred to as *niching* methods. A niching method can be incorporated into a standard Evolutionary Algorithm (EA) to promote and maintain the formation of multiple stable subpopulations within a single population, with an aim to locate multiple optimal solutions.

Most existing niching algorithms, however, require user specified parameters in order to perform well. In a recent attempt to eliminate the need of specifying any niching parameters, a $lbest$ PSO niching algorithm based on a ring topology was proposed[3], where a PSO's population was directly mapped onto a ring topology. This way the population can be naturally divided into multiple subpopulations, each operating as a separate PSO with its own local neighbourhood best. Since these subpopulations are only loosely connected, the speed of convergence is greatly reduced. Most importantly, it was shown 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 optima. This $lbest$ PSO niching algorithm was shown to provide better performance than some existing niching algorithms typically using a niche radius parameter.

One potential issue as a result of niching is that these subpopulations may contain very few particles. In the ring topology based PSO, each subpopulation contains only 3 particles. The search capability of such a small swarm may be adequate for very low dimensional problems (e.g., 1 or 2 dimensions), but severely limited for higher dimensional multimodal problems. To further enhance the search capability of small swarms, there is a need to incorporate a more capable optimizer in place of the existing constricted PSO. This paper examine several alternative PSO variants using different position update rules, and compare the effects of embedding these schemes in the aforementioned $lbest$ PSO niching algorithm using a ring topology. A novel PSO update rule which uses a combination of Cauchy and Gaussian distributions (CGPSO) to sample a particle's next position is proposed. The proposed CGPSO is shown to perform competitively even when a small population size is used. The $lbest$ PSO niching algorithm employing CGPSO seemed to be able to locate more global peaks other PSO variants on multimodal problems which have many global peaks but few or no local peaks.

The rest of the paper is organized as follows. We begin with a review of classic niching methods in section II, focusing on discussing the issue of niching parameters. We then give a brief introduction on PSO and several PSO variants based on Gaussian distribution in section III. In section IV we describe in detail a $lbest$ PSO niching algorithms proposed in [3]. This is followed by section V describing a newly proposed PSO update rule employing Cauchy and Gaussian distributions for sampling a particle's next position. Experimental setup and numerical results are presented in section VI and VII respectively. Finally section VIII gives the concluding remarks.

II. Niching Methods

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 [4]. One of the early niching methods was developed by De Jong in a scheme called *crowding*. In *crowding*, an offspring is compared to a small random sample taken from the current population, and the most similar individual in the sample is replaced. A parameter CF (*crowding factor*) is commonly used to determine the size of the sample. The most widely used niching method is probably *fitness sharing*. The sharing concept was originally introduced by [5], and then adopted

X. Li is with the School of Computer Science and IT, RMIT University, Melbourne, VIC 3001, Australia, (phone: +61 3 99259585; e-mail: xiaodong.li@rmit.edu.au).

K. Deb is with the Department of Mechanical Engineering, Indian Institute of Technology Kanpur, PIN 208016, India (phone: +91 512 2597205; email: deb@iitk.ac.in).

978-1-4244-8126-2/10/$26.00

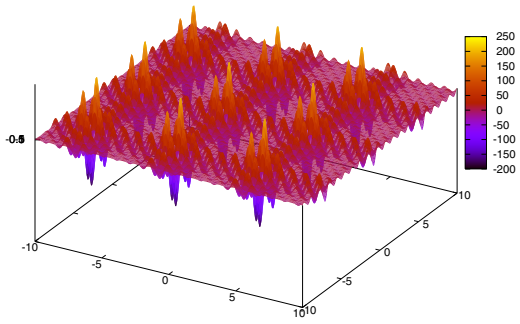


Fig. 1. Inverted Shubert 2D function.

by [2] as a mechanism to divide the population into different subpopulations according to the similarity of the individuals in the population. Fitness sharing was inspired by the *sharing* concept observed in nature, where an individual has only limited resources that must be shared with other individuals occupying the same niche in the environment. A sharing function is often used to degrade an individual's fitness based on the presence of other neighbouring individuals. Although *fitness sharing* has proven to be a useful niching method, it has been shown that there is no easy task to set a proper value for the sharing radius parameter σ_{share} in the sharing function without prior knowledge of the problems [6] .

Apart from the above, many other niching methods have also been developed over the years, including *derating* [7], *deterministic crowding* [8], *restricted tournament selection* [9], *parallelization* [10], *clustering* [11], and *speciation* [12], [13]. Niching methods have also been developed for PSOs, such as *NichePSO* [14] and *SPSO* [15].

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* [2], 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) [15]. The performance of these EAs depend very much on how these parameters are specified.

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

In the following section we will first present a brief introduction on PSO. This is then followed by section IV describing a PSO niching algorithm using a ring topology that is able to remove the need of specifying niching parameters.

III. Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a Swarm Intelligence technique originally developed from studies of social behaviours of animals or insects [16]. In a canonical PSO, the velocity of each particle is modified iteratively by its *personal best* position (i.e., the position it has attained that gives the best fitness value so far), and the *global best* position (i.e., 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 global best position. Let's use $\vec{\mathbf{v}}_i$ to denote the velocity of the i-th particle in the swarm, $\vec{\mathbf{x}}_i$ its position, $\vec{\mathbf{p}}_i$ its personal best, and $\vec{\mathbf{p}}_g$ the global best position from the entire swarm. $\vec{\mathbf{v}}_i$ and $\vec{\mathbf{x}}_i$ of the i-th particle in the swarm are updated according to the following two equations [17]:

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

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

where $\vec{\mathbf{R}}_1[0,\varphi_1]$ and $\vec{\mathbf{R}}_2[0,\varphi_2]$ are two separate functions each returning a vector comprising random values uniformly generated in the range [0, φ_1] and [0, φ_2] respectively. φ_1 and φ_2 are commonly set to $\frac{\varphi}{2}$ (where φ is a positive constant). The symbol $\otimes$ denotes point-wise vector multiplication. A constriction coefficient χ is used to prevent each particle from exploring too far away in the search space, since χ applies a dampening effect to the oscillation size of a particle over time. This Type 1" constricted PSO suggested by Clerc and Kennedy is often used with χ set to 0.7298, calculated according to $\chi = \frac{2}{\left|2-\varphi-\sqrt{\varphi^2-4\varphi}\right|}$, where $\varphi = \varphi_1 + \varphi_2 = 4.1$ [17].

A. *lbest and gbest PSO*

Two common approaches of choosing $\vec{\mathbf{p}}_g$ in equation (1) are known as $gbest$ and $lbest$ methods. In a $gbest$ PSO, the position of each particle in the search space is influenced by the best-fit particle in the entire population, whereas a $lbest$ PSO only allows each particle to be influenced by the best-fit particle chosen from its neighborhood. The $lbest$ PSO with a neighborhood size set to the population size is equivalent to a $gbest$ PSO. Kennedy and Mendes [18] studied PSOs with various population topologies. One of common population topologies suggested is the ring topology, where each particle

on the population array is only allowed to interact with its two immediate neighbours. A more recent study in [19] showed that a $lbest$ PSO is most likely to outperform a $gbest$ PSO on a multimodal fitness landscape, whereas $gbest$ PSO tends to do better on a unimodal landscape. Figure 2 illustrates a typical example using a ring topology for an EA and an equivalent for a PSO. One noticeable difference is that the PSO approach uses *local memory*.

Clearly the ring topology is desirable for locating multiple optima, because ideally we would like to have individuals to search thoroughly in its local neighbourhood before propagating the information throughout the population. The consequence of any quicker than necessary propagation would result in the population converging onto a single optimum (like $gbest$ PSO). The ring topology seems to be able to provide the right amount of communication needed for inducing stable niching behaviour.

B. Gaussian based PSO variants

It is interesting to note that if we let $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$ be fixed, then the resulting particle positions from (1) and (2) follow a Gaussian distribution [21]. This observation led to Kennedy proposing a PSO variant called Bare-bones PSO which relies solely on the Gaussian distribution to generate a particle's next position [21]. In the Bare-bones PSO, each dimension of the new position of the i-th particle is randomly selected from a Gaussian distribution with the mean being the average of $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$ and the standard deviation σ being the distance between $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$:

$$\vec{\mathbf{x}}_i \leftarrow \mathcal{N}(\frac{\vec{\mathbf{p}}_i + \vec{\mathbf{p}}_g}{2}, |\vec{\mathbf{p}}_i - \vec{\mathbf{p}}_g|). \quad (3)$$

Note that there is no velocity term used in equation (3). The new particle position is simply generated via the Gaussian distribution. A comparative study on PSO variants employing Gaussian distribution was provided in [22], as well as Lévy distribution which is a more generalized form of distribution than Gaussian and Cauchy distribution[1]. Algorithms employing Lévy or Cauchy distribution, which both have a long fat tail, are more capable of escaping from local optima than the Gaussian counterpart, as suggested in several studies [23], [24], [22].

Without the velocity term $\vec{\mathbf{v}}_i$, the Gaussian based PSO as shown in (3) becomes very similar to Evolutionary Programming (EP), where Gaussian distribution is typically used for generating the next trial points in the search space [23]. One important difference though, is that in the Gaussian based PSO the standard deviation σ is determined by the distance between $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$, whereas in a typical EP, σ needs to be supplied by the user, or be made self-adaptive [23], [24].

In another Gaussian based PSO (GPSO) proposed by Secrest and Lamont [25], instead of sampling around the midpoint between $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$, Gaussian distribution is used to sample around $\vec{\mathbf{p}}_g$ with some pre-specified probability p, otherwise around $\vec{\mathbf{p}}_i$. This proves to be beneficial, as particles can explore better in a much wider area, rather than just around the midpoint between $\vec{\mathbf{p}}_i$ and $\vec{\mathbf{p}}_g$. However, since GPSO uses only Gaussian distribution, its ability to explore the search space is still rather limited especially when the standard deviation becomes smaller. In section V we propose a PSO update rule which uses a combination of Cauchy and Gaussian distributions (instead of only Gaussian distribution) to sample a particle's next position. Our results show that this Cauchy and Gaussian based PSO with a small population size is able to explore more effectively the search space than GPSO and other PSO variants.

[1]The shape of the Lévy distribution can be controlled by a parameter α. For $\alpha = 2$ it is equivalent to Gaussian distribution, whereas for $\alpha = 1$ it is equivalent to Cauchy distribution [22].

```
Randomly generate an initial population
repeat
    for i = 1 to Population Size do
        if fit(x_i) > fit(p_i) then p_i ← x_i;
    end
    for i = 1 to Population Size do
        p_{n,i} ← neighbourhoodBest(p_{i-1}, p_i, p_{i+1});
    end
    for i = 1 to Population Size do
        Equation (1);
        Equation (2);
    end
until termination criterion is met;
```

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

IV. $lbest$ Niching PSO Using a Ring Topology

This section describes the $lbest$ PSO niching algorithm using a ring topology, which was proposed in [3]. As shown in Figure 2 b) or c), in a $lbest$ niching PSO using a ring topology, each particle interacts only with its immediate neighbours. An implementation of such a $lbest$ PSO using a ring topology is provided in Algorithm 1. Note that we can conveniently use population indices to identify the left and right neighbours of each particle. Here we assume a 'wrap-around' ring topology, i.e., the first particle is the neighbour of the last particle and vice versa. The $neighbourhoodBest()$ function returns the neighbourhood best of the i-th particle (which is the best-fit *personal best* in the i-th particle's neighbourhood). This neighbourhood best, i.e., $\vec{\mathbf{p}}_{n,i}$, is then used as the i-th particle's *local leader* (instead of $\vec{\mathbf{p}}_g$) when updating the i-th particle using equations (1) and (2).

Note that different particles residing on the ring can have different $\vec{\mathbf{p}}_n$ (we use $\vec{\mathbf{p}}_n$ to denote a non-specific 'neighbourhood best'), and they do not necessarily converge into a single value over time. As illustrated in Figure 3, the ring topology not only provides a mechanism to slow down information propagation in the particle population, but also allows different neighbourhood bests to *coexist* (rather

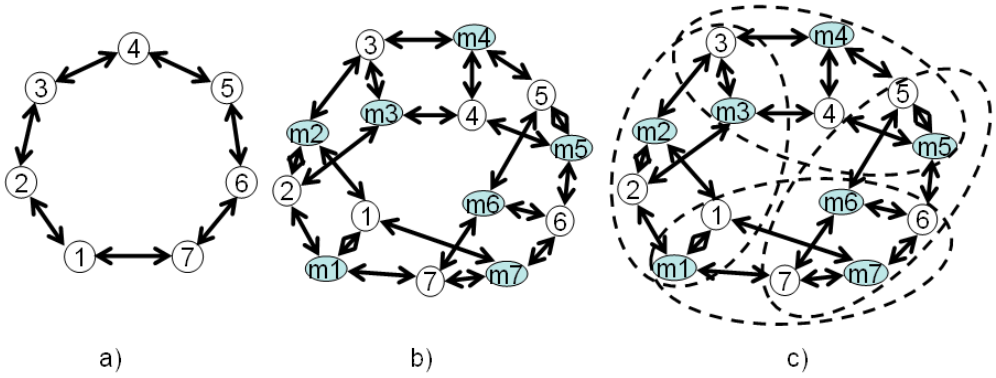


Fig. 2. a) A typical ring topology used in a conventional EA. Each member interacts only with its immediate left and right neighbours, with no *local memory* used; b) Graph of influence for a $lbest$ PSO using the same ring topology (see also p.89 in [20]). Each particle possesses a *local memory*; c) The same as b) but also showing the overlapping subpopulations, each consisting of a particle and its two immediate neighbours, and their corresponding memories.

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 neighbourhood, but not by a better $\vec{\mathbf{p}}_n$ of its neighbouring particle. Assuming that particles from the initial population are uniformly distributed across the search space, niches can naturally emerge as a result of the coexistence of multiple $\vec{\mathbf{p}}_n$ positions being the local attraction points for the particles in the population. With a reasonable population size, such a $lbest$ PSO is able to form stable niches around the identified neighbourhood bests $\vec{\mathbf{p}}_n$.

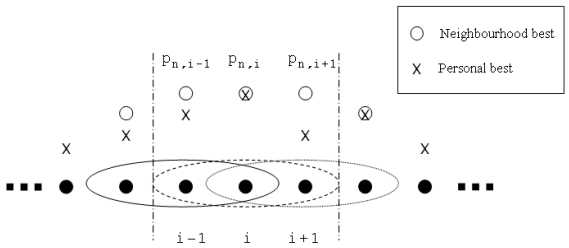


Fig. 3. A ring topology with each member interacting with its 2 immediate neighbours (left and right). Local neighbourhoods are overlapped with each other. The i-th particle's neighbourhood best ($\vec{\mathbf{p}}_{n,i}$) is the same as those of its 2 immediate neighbouring particles, but differs from those particles in the neighbourhoods further out.

Extensive experiments carried out in [3] showed 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 named as **r3pso** (with each member interacting with its immediate member on its left and right as shown in Figure 3) are able to locate multiple global optima, given a reasonably large population size, whereas the $lbest$ PSO algorithms with a *non-overlapping* ring topology can be used to locate global as well as local optima, especially for low dimensional problems. We used **r3pso** for all our experiments described in this paper.

V. Cauchy and Gaussian Distributions Based PSO

Inspired by the Bare-bones PSO and GPSO as described in section III-B, in this section, we propose a PSO update rule that employs a combination of Cauchy and Gaussian distributions for sampling a particle's next position. The update rule for each particle position can be now rewritten as follows:

$$\vec{\mathbf{x}}_i \leftarrow \begin{cases} \vec{\mathbf{p}}_i + \mathcal{C}(1)|\vec{\mathbf{p}}_i - \vec{\mathbf{p}}_{n,i}|, & \text{if } rand \leq p; \\ \vec{\mathbf{p}}_{n,i} + \mathcal{N}(0,1)|\vec{\mathbf{p}}_i - \vec{\mathbf{p}}_{n,i}| & \text{otherwise,} \end{cases} \quad (4)$$

where $\mathcal{C}(1)$ denotes a number that is generated following a Cauchy distribution, and in this case we need to set an "effective standard deviation" [22] for the Cauchy distribution, which is the same standard deviation value we would set for the equivalent Gaussian distribution, $|\vec{\mathbf{p}}_i - \vec{\mathbf{p}}_{n,i}|$; $rand$ is a random number generated uniformly from [0,1]; p is a user-specified probability value for Cauchy sampling to occur. Here $\vec{\mathbf{p}}_{n,i}$ denotes a local neighbourhood best for the i-th particle. Since a $lbest$ ring topology is used for defining local neighbourhood, $\vec{\mathbf{p}}_{n,i}$ (i.e., the best-fit particle), is chosen among all 3 particles including the current i-th particle and its immediate left and right neighbours (imagine that all particles are stored on a list that is indexed and wrapped-around). Since each particle may have a different $\vec{\mathbf{p}}_n$, the population is likely to remain diverse for a longer period. The chance of prematurely converging to a single global best is reduced, since multiple local neighbourhood bests are used, instead of a single population best $\vec{\mathbf{p}}_g$ as in the Bare-bones PSO (see equation (3)). Note that p can be simply set to 0.5 so that half of the time Cauchy is used to sample around the personal best $\vec{\mathbf{p}}_i$, while for the other half of the time Gaussian is used to sample around the neighbourhood best $\vec{\mathbf{p}}_{n,i}$. The idea is to use Cauchy distribution to explore more broadly around different personal best positions so that the swarm remains diverse, and at the same time to use less exploratory Gaussian to encourage convergence towards various neighbourhood best positions.

VI. Experimental Design

A. Two sets of experiments

We carried out two sets of experiments. In the first set of experiments, we want to evaluate the search capability of the 4 PSO variants using only a small population size. Most niching algorithms need to subdivide the population into many smaller subpopulations (or niches), with each specializing in locating a different global peak, therefore the search capability of a niche (often with a small population size) is critical to the overall performance of the niching algorithm. In the case of **r3pso**, we want to pick the best optimizer for a ring topology defined local neighbourhood (i.e., with a size of 3), hence we set the population size to 3. We chose to use the standard *Sphere* and *Rastrigin* functions of 2 to 100 dimensions for this comparative study. The idea is to evaluate the ability of each PSO variant to converge on a simple unimodal function, as well as on a multimodal function. The Rastrigin function was used to assess if an algorithm can overcome many of the local optima before reaching the global optimum. Each algorithm was run 50 times (with each run allowing 200,000 evaluations), and the mean and standard deviation of the best fitness were recorded.

In the second set of experiments, we are more interested in evaluating the niching capability of the abovementioned **r3pso** employing 4 different PSO update rules. We used 6 multimodal optimization test functions of different characteristics (see Table I) for this purpose. f_1 has 5 evenly spaced global maxima, whereas f_2 has 5 global maxima unevenly spaced. f_3 has 4 global peaks with 2 closer to each other than the other 2. There are no local peaks. f_4 has 2 global peaks as well as 2 local peaks. f_5 is the inverted Shubert function, as shown in Figure 1. f_5 2D function has 9 groups of global optima, with 2 very close global optima in each group. For n-dimensional Shubert function, there are $n \cdot 3^n$ global optima unevenly distributed. These global optima are divided into 3^n groups, with each group having n global optima being close to each other. For f_5 3D, there are 81 global optima in 27 groups; whereas for f_5 4D, there are 324 global optima in 81 groups. f_5 will pose a serious challenge to any niching algorithm relying on a fixed niche radius parameter. f_6 the inverted Vincent function has 6^n global peaks, but unlike the regular distances between global peaks in f_5, in f_6 global peaks have vastly different spacing between them. Furthermore, f_6 has no local peaks.

In the second set of experiments, we compared four different **r3pso** variants employing the following four PSO update rules:

- *Constricted PSO (CPSO)*: as proposed by Clerc and Kennedy in [17]. See equations (1) and (2).
- *Bare-bones PSO (BPSO)*: as introduced by Kennedy in [21]. See equation (3).
- *Gaussian PSO (GPSO)*: as proposed by Secrest and Lamont in [25].
- *Cauchy and Gaussian PSO (CGPSO)*: as proposed in this paper. See equation (4).

For **r3pso** if any particle's $\vec{\mathbf{x}}_i$ exceeds the boundary of the variable range, its position is reset to a value which is twice of the right (or left boundary) subtracting $\vec{\mathbf{x}}_i$. For both GPSO and CGPSO, the probability p was set to 0.5.

B. Performance Measures

To measure the performance of the $lbest$ niching PSO using different update rules, we first allow a user to specify a *level of accuracy* (typically $0 < \epsilon \leq 1$), i.e., how close the computed solutions to the known global peaks are. If the difference between a found solution and a known global peak is below the specified ϵ, and it is sufficiently different from other already found global peaks, then we can consider a new peak is found. For only the purpose of measuring performance, we make use of an algorithm for identifying species seeds [15], in order to check if a niching algorithm has located all known global peaks. Basically at the end of a run, this algorithm is invoked to first sort all individuals in the population in decreasing order of fitness values. With a pre-specified 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 2 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 and also sufficiently different solutions (i.e., species seeds) based on the pre-specified niche radius and a given list of all personal best positions. For the test functions in Table I, since the exact number of global peaks is known *a priori*, and also roughly the distance between 2 closest global peaks, a niching algorithm's performance can be measured according to two performance measures - 1) *success rate* or 2) *peak ratio*, before reaching a pre-specified maximal number of evaluations and accuracy threshold ϵ. Here, *success rate* measures a niching algorithm's performance in terms of the percentage of runs in which all global optima are successfully located, whereas *peak ratio* measures the percentage of peaks (i.e., optima) located out of the total number of known global peaks in a run. To do this, we only need to check species seeds, which are the dominant particles sufficiently different from each other. At the end of each run, we can determine if all global peaks are found by checking all species seeds to see if they are close enough to all known global peaks.

VII. Results and Discussion

A. Competent optimizers

As shown in Tables II and III, even with just a small population size of 3, CGPSO outperformed significantly other PSO variants on both the unimodal Sphere and the multimodal Rastrigin functions. In particular, CGPSO's scalability

TABLE I

TEST FUNCTIONS.

Name	**Test function**	**Range**	**Number of global peaks**
Equal Maxima [26]	$f_1(x) = sin^6(5\pi x)$.	$0 \leq$ x ≤ 1	5
Uneven Maxima [26]	$f_2(x) = sin^6(5\pi(x^{3/4} - 0.05))$.	$0 \leq$ x ≤ 1	5
Himmelblau's function [26]	$f_3(x, y) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2$.	-6 $\leq$ x ≤ 6	4
Six-Hump Camel Back [27]	$f_4(x, y) = -4[(4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2]$.	$-1.9 \leq x \leq 1.9$; $-1.1 \leq y \leq 1.1$	2
Inverted Shubert function [13]	$f_5(x_1, x_2, \ldots, x_n) = -\prod_{i=1}^{n} \sum_{j=1}^{5} jcos[(j + 1)x_i + j]$.	$-10 \leq x_i \leq 10$	$n \cdot 3^n$
Inverted Vincent function [28]	$f_6(\vec{\mathbf{x}}) = \frac{1}{n} \sum_{i=1}^{n} sin(10 \cdot log(x_i))$	$0.25 \leq x_i \leq 10$	6^n

TABLE II

THE RESULTS OF PSO VARIANTS USING 4 DIFFERENT UPDATE RULES OVER 50 RUNS (MEAN AND STANDARD DEVIATION), ON THE STANDARD SPHERE FUNCTION (THE RESULT OF THE BEST PERFORMING ALGORITHM IS HIGHLIGHTED IN BOLD).

Dimension	Constricted PSO (CPSO)	Bare-bones PSO (BPSO)	Gaussian PSO (GPSO)	Cauchy and Gaussian PSO (CGPSO)
2	**1.55E-13 (1.09E-12)**	4.26E+02 (934.69)	4.89E+02 (1148.49)	3.05E-4 (1.90E-03))
3	5.03E-02 (0.30)	1.04E+03 (1471.87)	5.82E+02 (1121.36)	**1.17E-18 (5.49E-18)**
5	1.30E+02 (350.94)	2.06E+03 (1949.06)	1.43E+03 (1566.20)	**0.00E+00 (0.00E+00)**
10	1.48E+03 (1855.46)	7.53E+03 (3871.42)	4.89E+03 (3046.55)	**0.00E+00 (0.00E+00)**
20	1.05E+04 (4881.15)	2.17E+04 (7431.87)	1.45E+04 (5814.24)	**7.22E-130 (5.10E-129)**
50	5.50E+04 (10629.20)	7.76E+04 (11861.87)	5.52E+04 (12790.46)	**7.10E+00 (40.80)**
100	1.54E+05 (16667.09)	1.82E+05 (20800.63)	1.51E+05 (20992.62)	**7.11E+02 (1635.91)**

TABLE III

THE RESULTS OF PSO VARIANTS USING 4 DIFFERENT UPDATE RULES OVER 50 RUNS (MEAN AND STANDARD DEVIATION), ON THE STANDARD RASTRIGIN FUNCTION (THE RESULT OF THE BEST PERFORMING ALGORITHM IS HIGHLIGHTED IN BOLD).

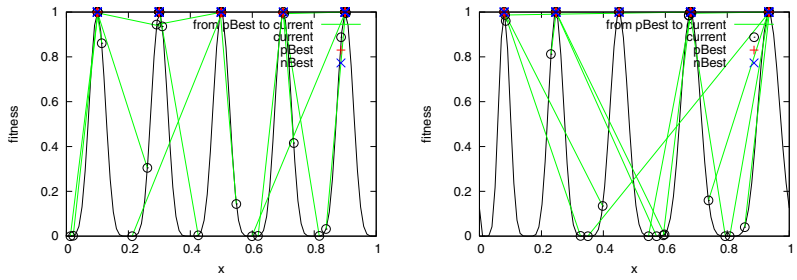
Dimension	Constricted PSO (CPSO)	Bare-bones PSO (BPSO)	Gaussian PSO (GPSO)	Cauchy and Gaussian PSO (CGPSO)
2	**1.75E+00 (2.77)**	3.39E+00 (4.16)	4.30E+00 (5.40)	2.81E+00 (4.32)
3	4.86E+00 (4.64)	8.46E+00 (5.81)	7.46E+00 (6.80)	**4.46E+00 (4.49)**
5	1.15E+01 (7.83)	1.79E+01 (10.42)	1.50E+01 (8.45)	**8.24E+00 (6.75)**
10	4.44E+01 (15.30)	4.87E+01 (15.57)	4.09E+01 (14.18)	**1.96E+01 (9.47)**
20	1.22E+02 (21.99)	1.46E+02 (26.64)	1.21E+02 (26.40)	**3.30E+01 (13.18)**
50	4.41E+02 (45.77)	4.90E+02 (50.37)	4.17E+02 (52.36)	**4.76E+01 (43.66)**
100	1.10E+03 (74.42)	1.17E+03 (72.93)	1.05E+03 (73.95)	**4.46E+02 (131.85)**

with increasing dimensions is far superior than any other PSO variants. Comparing GPSO and CGPSO, it is strikingly clear that using a combination of Cauchy and Gaussian distributions is far more effective than using only Gaussian distribution alone for sampling. These results provide a strong argument to employ CGPSO in the $lbest$ niching PSO algorithm, in order to further enhance its niching capability in particular on higher dimensional multimodal problems.

B. Different PSO update rules

Table V presents the success rates on f_1 to f_5. A population size of 50 was used for f_1 to f_4, and a population of 500 was used for f_5 inverted Shubert 2D, since it has 18 global peaks and numerous local peaks. The **r3pso** variants were run until all known global optima were found, or a maximum of 100,000 evaluations was reached. All results were averaged over 50 runs. In order to measure more accurately each **r3pso** niching variant's ability in forming niches in the vicinities of all known global optima, for f_1 to f_4, we set ϵ to 0.0001, and r to 0.01. In other words, a solution is found if it is less than $\epsilon = 0.0001$ from a known global peak's fitness, and at the same time not within the radius r of 0.01 in the search space of any other solutions already found. Table IV shows for each function the ϵ and r values used for performance measurement, estimated known global peak heights (which were obtained empirically), as well as the number of known global peaks (from Table I). Note that these values were purely used for the purpose of measuring performance over these test function, though in real-world scenarios we may not always have access to this sort of information.

As Table V shows, in terms of success rate, all 4 PSO variants performed similarly on f_1, f_2 and f_4, but both CPSO and CGPSO performed better than Bare-bones PSO and GPSO on f_3. f_3 has 2 global optima very close to each other, which may cause some difficulties to the Bare-bones PSO and GPSO. Figure 4 shows **r3pso** easily located all 5 global peaks on Deb's f_1 to f_2. Note that the convergence to the 5 peaks was stable, though a few particles always oscillated around. For the more challenging f_5 Inverted Shubert 2D (see Figure 1, a population size of 500 was used, and each



(a) f_1 (b) f_2

Fig. 4. Results of running **r3pso** (with a population size of 50) on f_1 to f_2 after 100 iterations. Note that $nBest$ denotes $\vec{\mathbf{p}}_n$, $pBest$ for $\vec{\mathbf{p}}_i$, and $current$ for $\vec{\mathbf{x}}_i$. A line is drawn from $\vec{\mathbf{p}}_i$ and its associated $\vec{\mathbf{x}}_i$.

TABLE IV
PARAMETERS USED FOR PERFORMANCE MEASUREMENT.

fnc	ϵ	r	Peak height	no. peaks
f_1	0.0001	0.01	1.0	5
f_2	0.0001	0.01	1.0	5
f_3	0.0001	0.01	200.0	4
f_4	0.0001	0.01	1.03163	2
f_5 (2D)	0.1	0.5	186.731	18
f_5 (3D)	0.1	0.5	2709.0935	81
f_5 (4D)	0.1	0.5	39303.55	324
f_6 (2D)	0.01	0.1	1.0	36
f_6 (3D)	0.01	0.1	1.0	216
f_6 (4D)	0.01	0.1	1.0	1296

TABLE V
SUCCESS RATES ON f_1 TO f_5 (1D OR 2D).

fnc	CPSO	Bare-bones PSO	GPSO	CGPSO
f_1	94%	**100%**	98%	96%
f_2	92%	90%	94%	**98%**
f_3	**54%**	24%	28%	44%
f_4	**100%**	98%	**100%**	**100%**
f_5 (2D)	**98%**	62%	76%	**98%**

algorithm was allowed to run for a maximum of 200,000 evaluations before termination. On f_5, Table V shows that both CPSO and CGPSO performed better than the other PSO variants. Figure 5 shows a series of snapshots of a typical run of $r3pso$ on f_5 2D. Multiple niches emerged from the run, which are clearly visible.

For f_5 3D, f_6 2D, and 3D, a population size of 500 was used, the maximal number of evaluations was set to 200,000. But for f_5 4D and f_6 4D, we doubled the population size to 1000 and allowed a maximum of 400,000 evaluations. Table VI shows the results on averaged peak ratio (instead of success rate), as most of the time the algorithms were unable to find all the peaks.

A general observation can be made from Table VI is that as the dimensionality increases (which also results in more global peaks present in the landscapes), the performances of all PSO variants decline quite dramatically. This suggests that there is clearly a need to design more competent niching

TABLE VI
AVERAGED PEAK RATIOS ON f_5 3D - 4D AND f_6 2D - 4D.

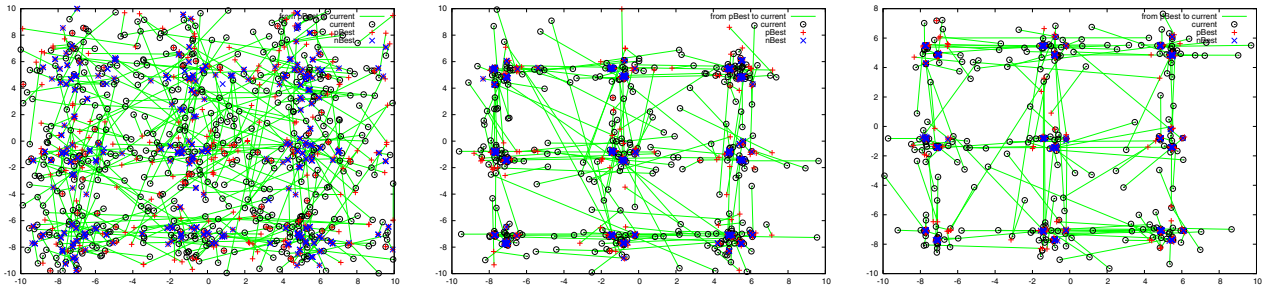
fnc	CPSO	Bare-bones PSO	GPSO	CGPSO
f_5 (3D)	**0.62**	0.40	0.45	0.31
f_5 (4D)	**0.24**	0.09	0.11	0.04
f_6 (2D)	0.95	0.77	0.80	**1.00**
f_6 (3D)	0.25	0.24	0.25	**0.35**
f_6 (4D)	0.10	0.10	0.10	**0.12**

algorithms that are more scalable. On these more challenging functions with dimensionality up to 4, we can also observe that CGPSO outperformed other variants including CPSO on f_6 2D, 3D and 4D, but did not on f_5 3D and 4D. f_6 Inverted Vincent 2D - 4D function have only global peaks present, whereas f_5 Inverted Shubert 3D and 4D function have 81 and 324 global peaks respectively, as well as numerous local peaks. This tells us that CGPSO still has the tendency of getting struck on local optima, while CPSO seems to fare much better in this regard.

Having said the above, CGPSO performed well on landscapes where there are many global peaks but very few or no local peaks. It seems that sampling using a combination of Cauchy and Gaussian distributions work better on this sort of fitness landscapes. A good example is on f_6 2D, where CGPSO was able to find all 36 peaks for all 50 runs (i.e., the averaged peak ratio is 1.0).

VIII. CONCLUSIONS

A niching EA algorithm typically subdivides a population into multiple niches, each operating as an independent EA with a relatively small population size. However, traditional EAs often do not work well with a small population size, therefore we argue the importance of employing a more competent optimizer in order for a niche to work well. This paper examines the effects of incorporating several different PSO update rules into a $lbest$ PSO niching algorithms. Among these variants, we also proposed a new PSO variant CGPSO which was shown empirically more competent than several other existing PSO variants, when a very small population size was used. By sampling using both Cauchy



(a) iteration 10 (b) iteration 50 (c) iteration 75

Fig. 5. The niching behaviour of the $r3pso$ (with a population size of 500) on the f_5 Inverted Shubert 2D function over a run.

and Gaussian distributions, CGPSO was shown to perform better than the more widely-used Constricted PSO (CPSO) on multimodal functions where there are many global peaks but fewer or no local peaks.

REFERENCES

[1] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems." Ph.D. dissertation, University of Michigan, 1975.

[2] D. E. Goldberg and J. Richardson, "Genetic algorithms with sharing for multimodal function optimization," in *Proc. of the Second International Conference on Genetic Algorithms*, J. Grefenstette, Ed., 1987, pp. 41–49.

[3] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. on Evol. Comput.*, vol. 14, no. 1, pp. 150 – 169, February 2010.

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

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

[6] D. E. Goldberg, K. Deb, and J. Horn, "Massive multimodality, deception, and genetic algorithms," in *PPSN 2*, R. Männer and B. Manderick, Eds. Amsterdam: Elsevier Science Publishers, B. V., 1992. [Online]. Available: citeseer.ist.psu.edu/goldberg92massive.html

[7] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evolutionary Computation*, vol. 1, no. 2, pp. 101–125, 1993. [Online]. Available: citeseer.ist.psu.edu/beasley93sequential.html

[8] S. W. Mahfoud, "Crowding and preselection revisited," in *Parallel problem solving from nature 2*, R. Männer and B. Manderick, Eds. Amsterdam: North-Holland, 1992, pp. 27–36. [Online]. Available: citeseer.ist.psu.edu/mahfoud92crowding.html

[9] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. of the Sixth International Conference on Genetic Algorithms*, L. Eshelman, Ed. San Francisco, CA: Morgan Kaufmann, 1995, pp. 24–31. [Online]. Available: citeseer.ist.psu.edu/harik95finding.html

[10] M. Bessaou, A. Pétrowski, and P. Siarry, "Island model cooperating with speciation for multimodal optimization," in *Parallel Problem Solving from Nature - PPSN VI 6th International Conference*, H.-P. S. et al., Ed. Paris, France: Springer Verlag, 16-20 2000. [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 *the International Conference on Artificial Neural Networks and Genetic Algorithms*, 1993, pp. 450–457.

[12] A. Pétrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. of the 3rd IEEE International Conference on Evolutionary Computation*, 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] R. Brits, A. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. of the 4th Asia-Pacific Conference on Simulated Evolution and Learning 2002(SEAL 2002)*, 2002, pp. 692–696.

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

[16] J. Kennedy and R. Eberhart, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[17] M. Clerc and J. Kennedy, "The particle swarm - explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. on Evol. Comput.*, vol. 6, pp. 58–73, Feb. 2002.

[18] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. of the 2002 Congress on Evolutionary Computation*, 2002, pp. 1671–1675.

[19] D. Bratton and J. Kennedy, "Defining a standard for particle swarm optimization," in *Proc. of 2007 IEEE Swarm Intelligence Symposium (SIS'07)*. IEEE, 2007, pp. 120 – 127.

[20] M. Clerc, *Particle Swarm Optimization*. London, UK: ISTE Ltd, 2006.

[21] J. Kennedy, "Bare bones particle swarm," in *Proc. of IEEE Int. Conf. Evolutionary Computation*, 2003, pp. 80–87.

[22] T. Richer and T. Blackwell, "The lévy particle swarm," in *Proc. of 2006 Congress on Evolutionary Computation (CEC'06)*. IEEE, 2006, pp. 808 – 815.

[23] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[24] C.-Y. Lee and X. Yao, "Evolutionary programming using mutations based on the lévy probability distribution," *IEEE Trans. on Evol. Comput.*, vol. 8, no. 1, pp. 1 – 13, Feb. 2004.

[25] B. Secrest and G. Lamont, "Visualizing particle swarm optimization - gaussian particle swarm optimization," in *Proc. of the 2003 IEEE Swarm Intelligence Symposium (SIS'03)*. IEEE, 2003, pp. 198 – 204.

[26] K. Deb, "Genetic algorithms in multimodal function optimization (master thesis and tcga report no. 89002)," Ph.D. dissertation, Tuscaloosa: University of Alabama, The Clearinghouse for Genetic Algorithms, 1989.

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

[28] O. Shir and T. Bäck, "Niche radius adaptation in the cms-es niching algorithm," in *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference (LNCS 4193)*. Reykjavik, Iceland: Springer, 2006, pp. 142 – 151.