A Distance-Based Locally Informed Particle Swarm Model for Multimodal Optimization

B. Y. Qu, Ponnuthurai Nagaratnam Suganthan, *Senior Member, IEEE,* and Swagatam Das, *Member, IEEE*

***Abstract*—Multimodal optimization amounts to finding multiple global and local optima (as opposed to a single solution) of a function, so that the user can have a better knowledge about different optimal solutions in the search space and when needed, the current solution may be switched to a more suitable one while still maintaining the optimal system performance. Niching particle swarm optimizers (PSOs) have been widely used by the evolutionary computation community for solving real-parameter multimodal optimization problems. However, most of the existing PSO-based niching algorithms are difficult to use in practice because of their poor local search ability and requirement of prior knowledge to specify certain niching parameters. This paper has addressed these issues by proposing a distance-based locally informed particle swarm (LIPS) optimizer, which eliminates the need to specify any niching parameter and enhance the fine search ability of PSO. Instead of using the *global best* particle, LIPS uses several *local bests* to guide the search of each particle. LIPS can operate as a stable niching algorithm by using the information provided by its neighborhoods. The neighborhoods are estimated in terms of Euclidean distance. The algorithm is compared with a number of state-of-the-art evolutionary multimodal optimizers on 30 commonly used multimodal benchmark functions. The experimental results suggest that the proposed technique is able to provide statistically superior and more consistent performance over the existing niching algorithms on the test functions, without incurring any severe computational burdens.**

***Index Terms*—Evolutionary computation, multimodal evolutionary optimization algorithm, niching technique, particle swarm optimization (PSO).**

I. Introduction

IN PRACTICAL optimization problems, it is often desirable to simultaneously locate multiple global and local optima of a given objective function. For real-world problems due to physical (and/or cost) constraints, the best results cannot always be realized. In such a scenario, if multiple solutions (local and global) are known, the implementation can be quickly switched to another solution while still maintaining an optimal system performance. Multiple solutions could also be analyzed to discover hidden properties (or relationships) of the concerned functional landscape. Thus, as the name suggests, a multimodal optimization task amounts to finding multiple optimal solutions and not just one single optimum, as it is done in a typical optimization study. If a point-by-point classical optimization approach is used for this task, the approach must be applied several times, every time hoping to find a different optimal solution.

Evolutionary algorithms (EAs) [1], [2], due to their population-based approach, provide a natural advantage over classical optimization techniques. They maintain a population of possible solutions, which are processed at every iteration, and if the multiple solutions can be preserved over all these iterations, then at termination of the algorithm we can have multiple good solutions, rather than only the best solution. Note that this is against the natural tendency of EAs, which will always tend to converge to the best solution or a suboptimal solution (in a rugged, not so well-posed function). Detection and maintenance of multiple solutions are two challenging tasks of using EAs to solve multimodal optimization problems.

Niching [3]–[7] refers to the technique of finding and preserving multiple stable niches, or favorable parts of the solution space possibly around multiple solutions, with a view of preventing convergence to a single solution. A niching method generally modifies the behavior of a classical EA in order to maintain multiple groups within a single population to locate different optima. Many niching methods have been developed over the years, including crowding [8], fitness sharing [9], restricted tournament selection (RTS) [10], and speciation [11]. Some of the classic niching techniques are presented in the next sections.

A. Crowding

Crowding [8] is one of the simplest and most commonly used niching techniques. It is inspired by the competition for limited resources among similar members of a natural population. The similarity is generally measured in terms of distance and one of the most common choices is the Euclidian distance. The crowding method is able to maintain the diversity of the whole population. The algorithm compares the offspring with some randomly sampled individuals from the current population. This approach requires a user-specified parameter called crowding factor (CF) to control the size of the sample. CF is generally chosen to be 2 or 3. The main

Manuscript received July 28, 2011; revised December 1, 2011 and February 20, 2012; accepted May 2, 2012. Date of publication June 6, 2012; date of current version May 24, 2013.

B. Y. Qu is with the School of Electric and Information Engineering, Zhongyuan University of Technology, Zhengzhou 450007, China (e-mail: E070088@ntu.du.sg).

P. N. Suganthan is with the School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore (e-mail: epnsugan@ntu.edu.sg).

S. Das is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata 700 108, India (e-mail: swagatamdas19 @yahoo.co.in).

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.2012.2203138

merit of crowding is its simplicity. However, it suffers from the replacement error.

B. Restricted Tournament Selection

RTS was originally introduced by Harik [10]. The concept is similar to crowding. RTS generates two offspring by applying crossover and mutation operators on two randomly selected parents from the current population. Then the method chooses a random sample of w (window size) individuals from the population and determines which one is the closest to the offspring. The closest member within the w individuals will compete with the offspring to determine the population for the next iteration.

C. Fitness Sharing

Fitness sharing is another effective multimodal optimization technique [9]. In fitness sharing, the population is divided into different subgroups according to the similarity of the individuals. Individuals share their information within the same subgroup. The shared fitness for ith individual can be represented by using the following equation:

$$f_{\text{shared}}(i) = \frac{f_{\text{original}}(i)}{\sum_{j=1}^{N} sh(d_{ij})}$$

where the sharing function is calculated using

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{\text{share}}}\right)^{\alpha}, & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{otherwise} \end{cases}$$

where d_{ij} is the distance between individuals i and j, σ_{share} is the sharing radius, N is the population size, and α is a constant called sharing level.

D. Speciation

Speciation [11] is based on the concept of separating the population into several species according to their similarity. Each of these species is formed around a dominating individual known as the species seed. The range of the species is controlled by a user-specified value called radius r_s. All individuals that fall within the radius from the species seed are identified as the same species.

The concept of particle swarms, although initially introduced for simulating the social behavior commonly observed in the animal kingdom, has become very popular as an efficient algorithm for intelligent search and optimization. Particle swarm optimization (PSO) [12]–[15], as it is called now, does not require any gradient information of the function to be optimized, uses only primitive mathematical operators, and is conceptually very simple. A few significant variants of PSO that have been used very effectively for solving various kinds of optimization problems can be found in [16]–[20].

The relative simplicity of PSO and the fact that it is a population-based technique has made it a natural candidate for solving multimodal optimization problems. Fitness Euclidean-distance ratio PSO (FERPSO) [21] and speciation-based PSO (SPSO) [22] are two commonly used and effective niching PSO algorithms. Recently, a ring topology PSO [23] has been used to overcome the niching parameter selection problem and solve multimodal problems. In this paper, a locally informed particle swarm (LIPS) optimizer algorithm is introduced to enhance local search ability and solve multimodal functions. Effectiveness of the proposed algorithm has been demonstrated by comparing its performance with nine other state-of-the-art multimodal optimizers over a test suite comprised of 15 basic and 15 composite multimodal benchmarks.

The remainder of this paper is organized in the following way. Section II gives a brief overview of the PSO algorithm, the niching techniques previously used along with PSO, and some other significant niching algorithms used in the comparative study undertaken here. In Section III, the LIPS algorithm is presented in sufficient detail. The problem definition and experimental results are presented and discussed in Sections IV and V, respectively. Finally, the paper concludes with Section VI.

II. Scientific Background and Related Works

A. Particle Swarm Optimization

PSO is a search technique that was originally introduced by Kennedy and Eberhart [13]. It has been shown to be effective in solving optimization problems [29], [32], [48], [49]. PSO emulates the group behavior of insects, birds flocking, and fish schooling, where these swarms search for food in a collective manner [14], [34]. In PSO dynamics, a swarm of particles (or agents), each representing a potential solution to the optimization problem at hand, navigates through the search space. The particles are initially distributed randomly over the feasible search space with a random velocity, and the goal is to converge to the global optimum of an objective function. Each particle keeps track of the best solution that it has achieved so far. This is the personal best value (the so-called pbest [14]). In addition, the PSO process also keeps track of the global best solution detected so far in a neighborhood of the current particle or in the entire swarm (the so-called gbest [14]). Thus, the velocity of each agent in the next iteration is computed by using the information of the gbest (as the social component), the best personal position of the particle pbest (as the cognitive component), and its current velocity (the memory term). Both social and cognitive components contribute randomly to the position of the agent in the next iteration.

The position X and velocity V of each particle are updated according to the following formulae:

$$V_i^d = \omega * V_i^d + c_1 * \text{rand1}_i^d * (\text{pbest}_i^d - X_i^d) + c_2 * \text{rand2}_i^d * (\text{gbest}^d - x_i^d) \quad (1)$$

$$X_i^d = X_i^d + V_i^d \quad (2)$$

where c_1 and c_2 are the acceleration constants and ω is the inertia weight to balance the global and local search performance. rand1_i^d and rand2_i^d are two random numbers within the range of [0, 1]. pbest_i is the best previous position yielding the best fitness value for the ith particle while gbest_i is the best position found by the entire swarm or some neighborhoods of the ith particle. Fig. 1 presents a typical flow chart of the conventional PSO.

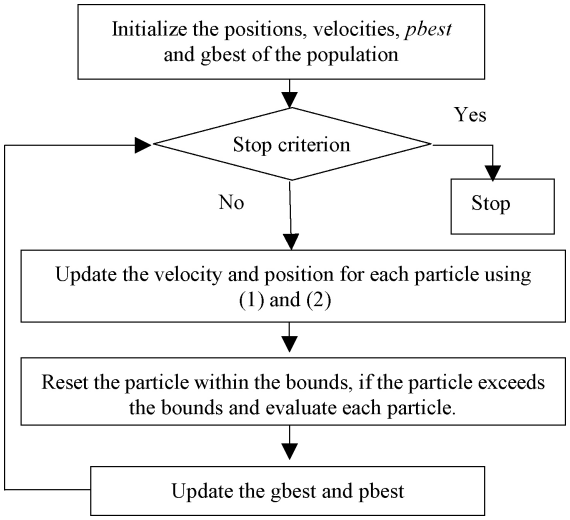

Fig. 1. Flow chart of a conventional PSO.

Algorithm 1 Pseudocode of an lbest PSO using a ring topology

```
Step 1   Randomly generate an initial population
         repeat
Step 2       For i=1 to population_size do
                 If f(X_i) > f(p_i) //compare the objective function
                                    values for maximization
                     Then p_i ← X_i
                 Endif
             Endfor
Step 3       For i=1 to population size do
                 p_{n,i} ← neighborhoodBest(p_{i-1}, p_i, p_{i+1}) //assign
                                                    neighborhood best
             Endfor
Step 4       For i=1 to population size do
                 Apply a standard lbest PSO
             Endfor
         Until termination criterion is met
```

B. Adapting PSO for Multimodal Optimization

For years, PSO has remained a favorite choice of researchers working on multimodal optimization problems. Parsopoulos *et al*. [24] used the objective function stretching as a sequential PSO niching technique, similar to that of Beasley *et al*. [25]. Once the PSO algorithm has identified a local maximum $f(X^*)$ (through comparing particle objective function values to a minimum threshold value), the objective function is stretched such that for each point X in the search space, if $f(X) > f(X^*)$, the point remains unaltered. But all other points, for which $f(X) \leq f(X^*)$ holds, are stretched so that X^* becomes a local minimum. All particles are then repositioned randomly. Thus, a potentially good solution is isolated once it is found and then the fitness landscape is stretched to keep other particles away from this area of the search space [26], [27]. In 2002, Brits *et al.* [28] proposed NichePSO, where multiple subswarms are grown from an initial swarm by monitoring the fitness of individual particles. The subswarms can merge together or absorb particles from the main swarm. NichePSO tracks the variance of a particle over iterations by monitoring its fitness. If the change in a particle's fitness over iterations is not very high, a subswarm is created with the particle's closest neighbor. The success of NichePSO depends on the proper initial distribution of particles throughout the search space. To ensure uniform distribution, *Faure* sequences [20] are used to initialize particle positions in the search space. The main swarm is trained using the cognition-only model [31] shown as

$$V_i^d = \omega * V_i^d + c_1 * \text{rand1}_i^d * (\text{pbest}_i^d - X_i^d). \tag{3}$$

In this model, only a conscience factor, in the form of a *personal best*, is considered when updating particle positions. Therefore, no social information in the form of a *global best* solution will influence position updates. In [33], Brits *et al.* proposed an nbest PSO model, where the neighborhood best of a particle is determined by taking the average of the positions of all particles in its neighborhood. Based on the Euclidean distance among the particles, the neighborhood of a particle can be defined by its k closest particles, k being a user-defined parameter.

In what follows, we discuss a few other significant niching PSOs that have been used in the comparative study undertaken in this paper.

1) *Fitness Euclidean-Distance Ratio PSO (FERPSO):* In FERPSO [21], instead of global best, the neighborhood best to each particle is used to lead the particles. Consequently, a solution moves toward its personal best as well as its fittest-and-closest neighbors (nbest), which are identified by the fitness-Euclidean distance ratio (FER) values. The nbest for ith particle is selected as the neighborhood personal best with the largest FER as follows:

$$\text{FER}_{(j,i)} = \alpha \cdot \frac{f(p_j) - f(p_i)}{\|p_j - p_i\|} \tag{4}$$

where $\alpha = \frac{\|s\|}{f(p_g) - f(p_w)}$ is a scaling factor, p_w is the worst-fit particle in the current population, p_j and p_i are the personal bests of the jth and ith particle, respectively, and $\|s\|$ is the size of the search space, which is estimated by its diagonal distance $\sqrt{\sum_{k=1}^{d}(x_k^u - x_k^l)^2}$ (where x_k^u and x_k^l are the upper and lower bounds of the kth dimension of the search space). The velocity update equation (1) in FERPSO is rewritten as

$$\begin{aligned} V_i^d = {} & \omega * V_i^d + c_1 * \text{rand1}_i^d * (\text{pbest}_i^d - X_i^d) \\ & + c_2 * \text{rand2}_i^d * (\text{nbest}^d - x_i^d). \end{aligned} \tag{5}$$

2) *Speciation-Based PSO (SPSO):* SPSO was first proposed by Li [22] in 2004. In SPSO, different subpopulations are formed as species, which are identified by the dominant particles known as the species seeds. To identify the species seeds and determine the size of species, a niching parameter known as radius must be specified by the user. The procedure for determining species and the species seeds was adopted from [35]. Each species and its corresponding species seed form a separate subpopulation that can be optimized with a PSO itself. Similar to FERPSO, SPSO replaces the global best

Algorithm 2 Steps of SDE

Step 1 Randomly generate NP number of initial trial solutions.

Step 2 Sort all individuals in descending order of their fitness values.

Step 3 Determine the species seeds for the current population; the most-fit individual will be set as the first species seed. Then all individuals are checked in turn from most-fit to least-fit against the species seeds found so far. If an individual does not fall in the radius of any seeds, it will be indentified as another species seed.

Step 4 For each species, execute a global DE variant.

4.1 If a species has less than m individuals, randomly generate new individuals within the radius of the species seed.

4.2 If the child's fitness is the same as its species seed, replace this child with a randomly generated new individual.

Step 5 Keep only the NP fitter individuals from the combined population.

Step 6 Stop if a termination criterion is satisfied. Otherwise go to Step 4.

Algorithm 3 Steps of CDE

Step 1 Randomly generate NP number of initial trial solutions.

Step 2 For $i = 1$ to NP

Produce an offspring u_i using the standard DE.

Calculate the Euclidean distance of u_i to the other individuals in the DE population.

Compare the fitness of u_i with the most similar individual and replace it if u_i has a better fitness value.

Endfor

Step 3 Stop if a termination criterion is satisfied. Otherwise go to Step 2.

by species best or species seed and all the particles in the same species at each iteration step share the same neighborhood best. Over successive iterations, multiple global optima can be found in parallel.

3) *Ring Topology PSO:* Recently, Li [23] proposed an lbest PSO with ring topology for niching. In ring topology PSO, each particle interacts only with its immediate neighbors. The algorithm makes use of ring topology to form different niches and realize multiple-peaks optimization, which does not require any niching parameters. Li showed that PSO algorithms using the ring topology were able to form stable niches across different local neighborhoods, eventually locating multiple global or local optima. The implementation of ring topology PSO is shown in Algorithm 1.

C. Other Niching Algorithms Compared in this Paper

This section presents brief overviews of three state-of-the-art multimodal algorithms (non-PSO based) compared in this paper. These algorithms are commonly used to exhibit competitive performances in multimodal optimization.

1) *Species-Based Differential Evolution:* Species-based differential evolution (SDE) [36] presents a commonly used niching algorithm incorporated in the framework of differential evolution (DE), a very powerful real parameter optimizer [37]. The concept is the same as SPSO described in Section II-B2. The steps of SDE are presented in Algorithm 2.

2) *Crowding Differential Evolution:* Crowding differential evolution (CDE) was first proposed by Thomsen [8]. In CDE, when an offspring is generated, it competes with the individual most similar to it (measured by Euclidean distance) in the current population. The offspring will replace this individual if it has a better fitness value. The steps of CDE are provided in Algorithm 3.

3) *Adaptive Niche Radii Covariance Matrix Adaptation Evolution Strategy:* Evolution strategies [38] are search procedures that mimic the natural evolution of the species in natural ecosystems. In [39], Shir *et al.* introduced an adaptive niche radii covariance matrix adaptation evolution strategy (CMA-ES) to solve multimodal problems. The algorithms solve the problem of selecting a suitable niching radius and offer an efficient niching mechanism with less presumption on the search landscape. This is successfully achieved at two levels: the construction of the self-adaptive niche radius and the application of the Mahalanobis distance metric for the adaptation of the niche shapes. The details of the algorithm can be found in [39].

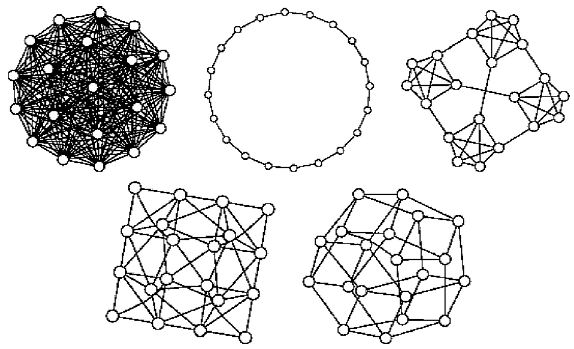

Fig. 2. Topologies used in FIPS [40].

III. Locally Informed PSO

A. Fully Informed Particle Swarm

In 2004, Mendes *et al.* [40] introduced a fully informed particle swarm (FIPS) to solve single global optimization problems. The classical particle swarm algorithm works by continuously searching in a region that is defined by each particle's best previous success, the best success of one neighborhood particle, the particle's current position, and its previous velocity [40]. The particle uses only one neighborhood best information to bias its search direction. However, there is no guarantee that the chosen neighborhood best will always lead to solutions better than other neighborhoods' bests. Important information contained in neighborhood particles may be neglected through overemphasis on the single best neighbor, which may lead to poor local search or slower convergence.

Unlike the canonical PSO, FIPS makes use of the information from all other particles around it, which is conceptually more concise and promises to perform better than the traditional particle swarm algorithm. In the FIPS, all neighbors are a source of influence in leading the particles to fly. Therefore, how to select the neighbors determines how diverse the influence will be and how efficient the algorithm will be. In [40], FIPS is integrated with five different PSO social

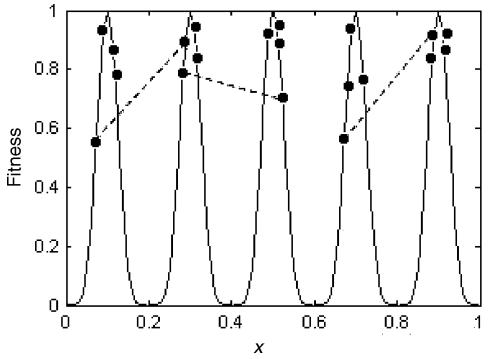

Fig. 3. Comparison between topology-based neighborhood and distance-based neighborhood.

networks and shows very promising performance in solving single global peak optimization problem. The five topologies are known as all, ring, four clusters, pyramid, and square, which are depicted in Fig. 2 [40]. The behavior of each particle is affected by its neighborhood identified by the topology used, which is able to make the whole population converge fast and more accurately.

B. Distance-Based Locally Informed Particle Swarm

Although FIPS is proven to be very effective in solving single-objective global optimization problems, it is not suitable for multimodal optimization due to the topology-based neighborhood selection method. As can be easily revealed, all the particles and their neighborhood are likely to converge to one point for single modal or single global peak optimization. However, for multimodal case, multiple peaks need to be located simultaneously and these peaks can be far from each other. Thus, if the topological neighborhood selection is used, the neighbors are likely to form from different areas or different niches. Take ring topology as an example; the particle is only allowed to interact with its immediate neighbors on its left and right according to the particle indices. The possibility for these neighbors to belong to different niches is very high as the initial population is randomly generated. If the neighbors are not from the same niche targeting a single peak, it is difficult for the niching algorithm to converge and locate the peaks effectively.

Motivated by this observation, a distance-based LIPS optimizer is presented in this paper. Similar to FIPS, besides using its personal best, LIPS adopts the local information from its nearest neighborhood (measured in terms of Euclidean distance) to guide the search of the particles. In this way, LIPS can form different stable niches that can converge to different global peaks. Fig. 3 highlights the benefits of Euclidean distance-based neighborhood over the topological neighborhood. In Fig. 3, there are five global optima that need to be located (F1: equal maxima [42]). The dots represent the particles of current population. The dashed lines connect topological neighbors according to particle indices. It cannot be avoided that some particles must have topological neighbors from different niches. As shown in Fig. 3, if two particles are from different niches, they may oscillate between two peaks that waste the function evaluations. Oscillation between two niches is unfavorable for exploration and exploitation. During the exploration stage (early search stage), particles are confined to locate additional niches that are positioned between two previously identified niches due to the oscillation. However, if there is no niche between two previously identified niches, oscillation wastes function evaluations. In contrast, the initial particles generated by the Euclidean distance-based neighborhood are well distributed over the whole search space and it facilitates the exploration process to be unrestricted. Hence, the proposed LIPS is more likely to perform exploration of the whole search space more freely than topological neighborhood-based PSO. During the exploitation stage (late search stage), oscillation in the topological neighborhood-based niching PSOs makes it hard to improve the accuracy while the proposed LIPS is able to perform search within each Euclidean distance-based neighborhood without any interference from other niche. Hence, LIPS is highly effective for fine search and thereby successfully locates the desired peaks with high accuracy.

Algorithm 4 Steps of local search

Step 1	Randomly generate the initial solutions.
Step 2	Evaluate the initial solutions and initialize the pbest.
	For i=1 to NP (population size)
Step 3	Identify the nearest (measured in Euclidean distance) *nsize* number neighborhood best to ith particle's pbest.
Step 4	Update the particles velocity using (6).
Step 5	Update the particles position using (2).
Step 6	Evaluate the newly generated particle.
Step 7	Update the pbest for the ith particle.
	Endfor
Step 8	Stop if a termination criterion is satisfied. Otherwise go to Step 3.

The velocity update of LIPS uses the formula given below while the position update keeps unchanged [40]

$$V_i^d = \omega \times (V_i^d + \varphi(P_i^d - X_i^d)) \tag{6}$$

where

$$P_i = \frac{\sum_{j=1}^{\text{nsize}} (\varphi_j \cdot \text{nbest}_j)/\text{nsize}}{\varphi}.$$

φ_j is a uniformly distributed random number in the range of [0, 4.1/nsize] and φ is equal to the summation of φ_j. nbest$j$ is the jth nearest neighborhood to ith particle's pbest. nsize is the neighborhood size, the impact of which will be discussed in sufficient detail in Section V. In this paper, nsize is dynamically increased from 2 to 5 over the function evaluations. There are two main advantages of LIPS to solve multimodal optimization problems as follows:

1) the benefit of FIPS velocity update equation to ensure good usage of all neighborhood information especially during the later stages of the search process and this leads to fast convergence and a high accuracy;
2) Euclidean distance-based neighborhood selection to ensure the neighbors are from the same niche and this increases the algorithm's ability for local search and fine-tuning.

With these two advantages, LIPS can easily find most of the peaks and maintain them until the end of the predefined budget

TABLE I
TEST FUNCTIONS

Test Function Part 1		Test Function Part 2 [41]	
Test Function Name/Dimensions	No. of Global Peaks	Test Function Name/ Dimensions	No. of Global Peaks
F1: Equal maxima/1-D [42]	5	F16: CF 1/10-D	8
F2: Decreasing maxima/1-D [42]	1	F17: CF 2/10-D	6
F3: Uneven maxima/1-D [42]	5	F18: CF 3/10-D	6
F4: Uneven decreasing maxima/1-D [42]	1	F19: CF 4/10-D	6
F5: Himmelblau's function/2-D [42]	4	F20: CF 5/10-D	6
F6: Six-hump camel back/2-D [43]	2	F21: CF 6/10-D	6
F7: Shekel's foxholes/2-D [44]	1	F22: CF 7/10-D	6
F8: Inverted Shubert function/2-D [35]	18	F23: CF 8/10-D	6
F9: Waves/2-D [45]	10	F24: CF 9/10-D	6
F10: Sphere/10-D	1	F25: CF 10/10-D	6
F11: Branin RCOS/2-D [35]	3	F26: CF 11/10-D	8
F12: Ackley/2-D [8]	1	F27: CF 12/10-D	8
F13: Michalewicz/2-D [8]	1	F28: CF 13/10-D	10
F14: Ursem F1/2-D [45]	1	F29: CF 14/10-D	10
F15: Ursem F3/2-D [45]	1	F30: CF 15/10-D	10

of the function evaluations for multimodal optimization. The details of LIPS algorithm are shown in Algorithm 4 in the form of a pseudocode. Note that the complexity of Euclidean distance-based neighborhood selection is $O(Np^2)$, where Np is the number of particles of the current population.

IV. EXPERIMENTAL SETUP

A. Experimental Setting

All the algorithms are implemented using MATLAB 7.1 and executed on the computer with an Intel Pentium 4 CPU and 2 GB of memory. The operating system is Microsoft Windows XP. The PSO and CMA-ES parameters used in this paper are adopted from [40] and [39], respectively, while the DE parameters are $F = 0.9$ and CR = 0.1. In total, 10 different multimodal algorithms are examined through our experiments.

1) LIPS: the locally informed PSO;
2) *r*2pso [23]: an lbest PSO with a ring topology; each member interacts with only its immediate member to its right;
3) *r*3pso [23]: an lbest PSO with a ring topology; each member interacts with its immediate member on its left and right;
4) *r*2pso-*lhc* [23]: same as *r*2pso, but with no overlapping neighborhoods;
5) *r*3pso-*lhc* [23]: same as *r*3pso, but with no overlapping neighborhoods;
6) SPSO [22]: speciation-based PSO;
7) FERPSO [21]: fitness-Euclidean distance ratio PSO;
8) SDE [36]: speciation-based DE;
9) CDE [8]: the original crowding DE;
10) SCMA-ES [22]: self-adaptive niching CMA-ES.

B. Test Functions

In experiment one, 30 benchmark multimodal test functions are used. These functions can be divided into two parts. The first 15 functions are classical test functions that were taken from different published papers. The remaining 15 functions are composition functions [41]. Different from most traditional test functions, global optima of the composition functions do not always reside at the origin or in the center of the search area. Moreover, the composition functions come with search space rotation, a feature that makes them even more difficult to solve. Basic information of the test functions are summarized in Table I. For functions F2, F7, F9, F11, F13, F14, and F15, the target is to locate all the peaks (global and local), while for the rest the objective is to search for the global optimum or optima and to escape the local peaks. All functions are considered for maximization; therefore, when the definitions are given for minimization, the functions are just sign-reversed.

TABLE II
TEST FUNCTION SETTING

Function No.	ε	r	Population Size	No. of Function Evaluations
F1	0.000001	0.01	50	10 000
F2	0.000001	0.01	50	10 000
F3	0.000001	0.01	50	10 000
F4	0.000001	0.01	50	10 000
F5	0.0005	0.5	50	10 000
F6	0.000001	0.5	50	10 000
F7	0.00001	0.5	50	10 000
F8	0.05	0.5	250	100 000
F9	0.001	0.2	200	30 000
F10	0.01	0.2	50	12 500
F11	0.001	0.5	200	20 000
F12	0.01	0.5	100	10 000
F13	0.0001	0.5	100	10 000
F14	0.000001	0.5	100	10 000
F15	0.00001	0.5	100	20 000
F16–F30	0.5	1	600	300 000

TABLE III
SUCCESS RATES AND RANKS (IN PARENTHESES) FOR TEST FUNCTIONS F1−F15

Test Function	LIPS	*r*2pso	*r*3pso	*r*2pso-*lhc*	*r*3pso-*lhc*	SPSO	FERPSO	SDE	CDE	SACMA-ES
F1	**1 (1)**	0.92 (3.5)	0.88 (5.5)	**1 (1)**	0.92 (3.5)	0.88 (5.5)	0.84 (7)	0.72 (8)	0.28 (9)	0 (10)
F2	0.96 (2)	0 (8)	0 (8)	0.64 (3)	0.04 (5)	**1 (1)**	0 (8)	0 (8)	0.48 (4)	0 (8)
F3	**1 (1)**	0.88 (6)	0.72 (7)	0.92 (4)	0.92 (4)	0.92 (4)	**1 (1)**	0.6 (8)	0.28 (9)	0 (10)
F4	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	**1 (1)**	0.96 (10)
F5	**1 (1)**	0.28 (5.5)	0.24 (7.5)	0.28 (5.5)	0.24 (7.5)	0 (9.5)	0.72 (2.5)	0.72 (2.5)	0 (9.5)	0.44 (4)
F6	**1 (1)**	0.56 (6.5)	0.6 (5)	0.56 (6.5)	0.52 (8)	0 (9.5)	0.96 (4)	**1 (1)**	0 (9.5)	**1 (1)**
F7	**1 (1)**	0.6 (5)	0.52 (6)	0.84 (3)	0.76 (4)	0.92 (2)	0 (8.5)	0 (8.5)	0 (8.5)	0 (8.5)
F8	**0.84 (1)**	0.04 (6)	0.04 (6)	0.04 (6)	0.2 (4)	0 (9)	0.52 (3)	0 (9)	0.72 (2)	0 (9)
F9	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**	**0 (1)**
F10	**1 (1)**	0 (7)	0 (7)	0 (7)	0 (7)	0 (7)	0 (7)	0 (7)	**1 (1)**	**1 (1)**
F11	**1 (1)**	0.8 (6)	0.8 (6)	0.76 (8)	0.8 (6)	0.64 (9)	**1 (1)**	**1 (1)**	0.08 (10)	**1 (1)**
F12	**1 (1)**	0.72 (6.5)	0.88 (5)	0.56 (8)	0.72 (6.5)	0.08 (9)	**1 (1)**	0.96 (4)	0 (10)	**1 (1)**
F13	**1 (1)**	0.96 (5.5)	**1 (1)**	0.92 (7)	0.96 (5.5)	0 (10)	0.04 (9)	**1 (1)**	**1 (1)**	0.44 (8)
F14	**1 (1)**	0.12 (4)	0 (7.5)	0.76 (2)	0.6 (3)	0 (7.5)	0 (7.5)	0 (7.5)	0 (7.5)	0 (7.5)
F15	**1 (1)**	0 (6)	0 (6)	0 (6)	0 (6)	0 (6)	0 (6)	0 (6)	0 (6)	0 (6)
Total rank	**16**	77.5	79.5	69	72	91	67.5	73.5	89	86

C. Population Size and Maximal Number of Evaluations

In this experiment, a level of accuracy (typically $0<\varepsilon<1$), indicating how close the computed solutions to the known global peaks are, needs to be specified in order to compare different algorithms. If the difference from a computed solution to a known global optimum is below ε, then the peak is considered to have been found. The level of accuracy (ε), niching radius (r), population size, and maximal number of function evaluations allowed are listed in Table II (these settings apply to all compared algorithms). Generally, population size and maximal number of function evaluations are related to the number of optima to be located. Large number of optima requires a larger population size and more function evaluations. Note that the performance measure of each algorithm depends on the specified level of accuracy.

D. Performance Measure

To compare the performance of different multimodal optimizers, 25 independent runs of each of the algorithms are taken on each problem. The performance of all multimodal algorithms is measured in terms of the following two criteria:

1) success rate (the percentage of runs in which all the global peaks are successfully located);
2) peak ratio [8] (the percentage of successfully located peaks).

Note that for more challenging functions with a tight level of accuracy, if all peaks cannot be found even in a single run, then the corresponding success rate will become zero.

V. EXPERIMENTAL RESULTS

This section presents and duly discusses the experimental results of our comparative study. All algorithms were run until all known peaks were found or the maximum number of function evaluations was exhausted. In Section V-E, the effect of the neighborhood size of LIPS is also discussed.

A. Success Rate

The success rates for all the algorithms on test functions F1–F15 are recorded and presented in Table III. The ranks of each algorithm are shown in parentheses while the total ranks (summation of all the individual ranks) are listed in the last row of the table. As can been seen from this table, LIPS achieves a much higher success rate than other niching algorithms on most of these 15 test functions. It also ranks the best if compared to the overall performance (total rank). The better performance is due to the better fine search generated by the locally informed particles. As LIPS selects several distance-based neighbors, the search can be easily localized with respect to each niche during the later stages of search. Local neighborhood concept is also able to form stable niches around different peaks, which is suitable for multimodal optimization.

Test functions F16–F30 are much more complex than functions F1–F15. For these test problems, no algorithm is able to generate a nonzero success rate. Therefore, peak ratio is used as the sole criterion and presented in Section V-B.

B. Peak Ratio

Peak ratio is another important criterion for comparing different niching algorithms. The results for test functions F1–F30 are shown in Tables IV and V. The best algorithm is highlighted in boldface for each test function. As can be seen, LIPS performs the best on the test functions, especially on F16–F30. As explained earlier, these functions are more challenging functions that require a fast (high convergence speed) and accuracy (good fine-tuning ability) from the optimization algorithm. For topological neighbor-based PSO (such as *r*2pso), their convergence speed and fine-tuning ability are adversely affected by topological neighborhood property. Although they can roughly find certain niches, they are not able to bring down the accuracy to a required level. Again this is due to the oscillation problem caused by their topological neighborhood. For LIPS, the distance-based neighborhood ensures a high diversity and high-speed search process for these multimodal problems. Especially at late search stage,

TABLE IV
PEAK RATIO (TEST FUNCTIONS F1−F15)

Test Function		LIPS	*r*2pso	*r*3pso	*r*2pso-*lhc*	*r*3pso-*lhc*	SPSO	FERPSO	SDE	CDE	SACMA-ES
F1	Worst	1.00	0.80	0.80	1.00	0.80	0.80	0.80	0.80	0.20	0.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.20
	Mean	**1.00**	0.98	0.98	**1.00**	0.98	0.98	0.97	0.94	0.77	0.01
	Std	0.00	0.06	0.07	0.00	0.06	0.07	0.07	0.09	0.21	0.04
	t-test	–	0	0	0	0	0	0	1	1	1
F2	Worst	0.80	0.20	0.20	0.60	0.20	1.00	0.20	0.20	0.40	0.20
	Best	1.00	0.20	0.20	1.00	1.00	1.00	0.20	0.60	1.00	0.20
	Mean	0.99	0.20	0.20	0.90	0.56	**1.00**	0.20	0.30	0.86	0.20
	Std	0.04	0.00	0.00	0.14	0.22	0.00	0.00	0.12	0.17	0.00
	t-test	–	1	1	1	1	0	1	1	1	1
F3	Worst	1.00	0.80	0.80	0.80	0.60	0.80	1.00	0.80	0.40	0.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
	Mean	**1.00**	0.98	0.94	0.98	0.98	0.98	**1.00**	0.92	0.79	0.00
	Std	0.00	0.07	0.09	0.06	0.09	0.06	0.00	0.10	0.18	0.00
	t-test	–	0	1	0	0	0	0	1	1	1
F4	Worst	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mean	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.96
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.20
	t-test	–	0	0	0	0	0	0	0	0	0
F5	Worst	1.00	0.25	0.25	0.00	0.25	0.00	0.50	0.75	0.00	0.75
	Best	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	0.25	1.00
	Mean	**1.00**	0.73	0.69	0.75	0.78	0.21	0.92	0.93	0.08	0.85
	Std	0.00	0.22	0.22	0.23	0.17	0.16	0.14	0.12	0.12	0.13
	t-test	–	1	1	1	1	1	1	1	1	1
F6	Worst	1.00	0.00	0.00	0.50	0.00	0.00	0.50	1.00	0.00	1.00
	Best	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	0.50	1.00
	Mean	**1.00**	0.72	0.78	0.78	0.74	0.04	0.98	**1.00**	0.02	**1.00**
	Std	0.00	0.36	0.29	0.26	0.30	0.14	0.10	0.00	0.10	0.00
	t-test	–	1	1	1	1	1	0	0	1	0
F7	Worst	1.00	0.92	0.88	0.96	0.92	0.96	0.12	0.04	0.28	0.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	0.28	0.08	0.80	0.16
	Mean	**1.00**	0.98	0.97	0.99	0.98	**1.00**	0.21	0.05	0.50	0.04
	Std	0.00	0.03	0.03	0.01	0.03	0.01	0.05	0.02	0.10	0.03
	t-test	–	1	1	0	1	0	1	1	1	1
F8	Worst	0.94	0.67	0.72	0.67	0.72	0.28	0.83	0.50	0.89	0.00
	Best	1.00	1.00	1.00	1.00	1.00	0.72	1.00	0.94	1.00	0.28
	Mean	**0.99**	0.84	0.86	0.84	0.90	0.47	0.97	0.69	0.98	0.12
	Std	0.02	0.09	0.07	0.07	0.07	0.13	0.04	0.11	0.03	0.07
	t-test	–	1	1	1	1	1	0	1	0	1
F9	Worst	0.50	0.30	0.30	0.30	0.30	0.10	0.10	0.10	0.40	0.10
	Best	0.70	0.50	0.50	0.60	0.50	0.50	0.20	0.50	0.80	0.30
	Mean	**0.60**	0.37	0.37	0.40	0.38	0.28	0.11	0.28	0.59	0.22
	Std	0.06	0.07	0.06	0.08	0.07	0.11	0.03	0.08	0.10	0.09
	t-test	–	1	1	1	1	1	1	1	0	1
F10	Worst	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
	Best	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
	Mean	**1.00**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**1.00**	**1.00**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	0	0
F11	Worst	1.00	0.67	0.67	0.67	0.67	0.67	1.00	1.00	0.00	1.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	Mean	**1.00**	0.93	0.93	0.92	0.93	0.88	**1.00**	**1.00**	0.36	**1.00**
	Std	0.00	0.14	0.14	0.15	0.14	0.16	0.00	0.00	0.30	0.00
	t-test	–	1	1	1	1	1	0	0	1	0
F12	Worst	1.00	0.00	0.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00
	Best	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	1.00
	Mean	**1.00**	0.72	0.88	0.56	0.72	0.08	**1.00**	0.96	0.00	**1.00**
	Std	0.00	0.46	0.33	0.51	0.46	0.28	0.00	0.20	0.00	0.00
	t-test	–	1	0	1	1	1	0	0	1	0
F13	Worst	1.00	0.50	1.00	0.50	0.50	0.00	0.50	1.00	1.00	0.50
	Best	1.00	1.00	1.00	1.00	1.00	0.50	1.00	1.00	1.00	1.00
	Mean	**1.00**	0.98	**1.00**	0.96	0.98	0.16	0.52	**1.00**	**1.00**	0.72
	Std	0.00	0.10	0.00	0.14	0.10	0.24	0.10	0.00	0.00	0.26
	t-test	–	0	1	0	0	1	1	0	0	1
F14	Worst	1.00	0.00	0.50	0.50	0.50	0.00	0.50	0.50	0.00	0.50
	Best	1.00	1.00	0.50	1.00	1.00	0.00	0.50	0.50	0.50	0.50
	Mean	**1.00**	0.52	0.50	0.88	0.80	0.00	0.50	0.50	0.08	0.50
	Std	0.00	0.23	0.00	0.22	0.25	0.00	0.00	0.00	0.19	0.00
	t-test	–	1	1	1	1	1	1	1	1	1
F15	Worst	1.00	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.00	0.00
	Best	1.00	0.40	0.20	0.40	0.80	0.00	0.20	0.60	0.20	0.60
	Mean	**1.00**	0.10	0.07	0.18	0.16	0.00	0.20	0.29	0.01	0.48
	Std	0.00	0.13	0.10	0.13	0.21	0.00	0.00	0.16	0.04	0.18
	t-test	–	1	1	1	1	1	1	1	1	1
t-test summary	Better	–	11	12	10	11	10	8	10	10	10
	Similar	–	4	3	5	4	5	7	5	5	5
	Worst	–	0	0	0	0	0	0	0	0	0

TABLE V
PEAK RATIO (TEST FUNCTIONS F16−F30)

Test Function		LIPS	*r*2pso	*r*3pso	*r*2pso-*lhc*	*r*3pso-*lhc*	SPSO	FERPSO	SDE	CDE	SACMA-ES
F16	Worst	0.38	0.00	0.00	0.00	0.00	0.00	0.13	0.13	0.00	0.25
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.25	0.25	0.00	0.25
	Mean	**0.46**	0.00	0.00	0.00	0.00	0.00	0.14	0.22	0.00	0.25
	Std	0.06	0.00	0.00	0.00	0.00	0.00	0.04	0.05	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	1
F17	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.33	0.17	0.17	0.17
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.33	0.33
	Mean	**0.35**	0.00	0.00	0.00	0.00	0.00	0.33	0.20	0.20	0.32
	Std	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.07	0.05
	t-test	–	1	1	1	1	1	0	1	1	0
F18	Worst	0.67	0.00	0.00	0.00	0.00	0.00	0.33	0.17	0.00	0.33
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.50	0.33	0.17	0.50
	Mean	**0.67**	0.00	0.00	0.00	0.00	0.00	0.42	0.25	0.12	0.45
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.09	0.08	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F19	Worst	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17
	Mean	**0.32**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.03
	Std	0.15	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07
	t-test	–	1	1	1	1	1	1	1	1	1
F20	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.33	0.17	0.17	0.17
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.33	0.33
	Mean	**0.37**	0.00	0.00	0.00	0.00	0.00	0.33	0.22	0.18	0.32
	Std	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.05	0.05
	t-test	–	1	1	1	1	1	0	1	1	0
F21	Worst	0.50	0.00	0.00	0.00	0.00	0.00	0.17	0.17	0.00	0.33
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.50
	Mean	**0.62**	0.00	0.00	0.00	0.00	0.00	0.20	0.23	0.00	0.43
	Std	0.08	0.00	0.00	0.00	0.00	0.00	0.07	0.09	0.00	0.09
	t-test	–	1	1	1	1	1	1	1	1	1
F22	Worst	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.17
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.17	0.33	0.00	0.17
	Mean	**0.67**	0.00	0.00	0.00	0.00	0.00	0.08	0.17	0.00	0.17
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.14	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	1
F23	Worst	0.50	0.00	0.00	0.00	0.00	0.00	0.17	0.17	0.00	0.33
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.50
	Mean	**0.53**	0.00	0.00	0.00	0.00	0.00	0.25	0.23	0.00	0.38
	Std	0.07	0.00	0.00	0.00	0.00	0.00	0.09	0.09	0.00	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F24	Worst	0.50	0.00	0.00	0.00	0.00	0.00	0.17	0.17	0.00	0.17
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.33	0.50	0.00	0.33
	Mean	**0.58**	0.00	0.00	0.00	0.00	0.00	0.25	0.30	0.00	0.28
	Std	0.09	0.00	0.00	0.00	0.00	0.00	0.09	0.11	0.00	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F25	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.17	0.17	0.00	0.17
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.33	0.33	0.00	0.33
	Mean	**0.35**	0.00	0.00	0.00	0.00	0.00	0.18	0.18	0.00	0.23
	Std	0.05	0.00	0.00	0.00	0.00	0.00	0.05	0.05	0.00	0.09
	t-test	–	1	1	1	1	1	1	1	1	1
F26	Worst	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.00
	Best	0.83	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.17
	Mean	**0.68**	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	0.12
	Std	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.00	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F27	Worst	0.50	0.00	0.00	0.00	0.00	0.00	0.17	0.17	0.00	0.17
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.33	0.50	0.00	0.33
	Mean	**0.58**	0.00	0.00	0.00	0.00	0.00	0.27	0.27	0.00	0.28
	Std	0.09	0.00	0.00	0.00	0.00	0.00	0.09	0.14	0.00	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F28	Worst	0.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	0.40	0.00	0.00	0.00	0.00	0.00	0.10	0.20	0.00	0.30
	Mean	**0.40**	0.00	0.00	0.00	0.00	0.00	0.03	0.09	0.00	0.14
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.06	0.00	0.12
	t-test	–	1	1	1	1	1	1	1	1	1
F29	Worst	0.10	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.00	0.10
	Best	0.10	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.00	0.10
	Mean	**0.10**	0.00	0.00	0.00	0.00	0.00	**0.10**	**0.10**	0.00	**0.10**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	0	0	1	0
F30	Worst	0.40	0.00	0.00	0.00	0.00	0.00	0.10	0.10	0.00	0.10
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.20	0.20	0.00	0.30
	Mean	**0.48**	0.00	0.00	0.00	0.00	0.00	0.12	0.16	0.00	0.20
	Std	0.04	0.00	0.00	0.00	0.00	0.00	0.04	0.05	0.00	0.07
	t-test	–	1	1	1	1	1	1	1	1	1
t-test summary	Better	–	15	15	15	15	15	12	14	15	12
	Similar	–	0	0	0	0	0	3	1	0	3
	Worst	–	0	0	0	0	0	0	0	0	0

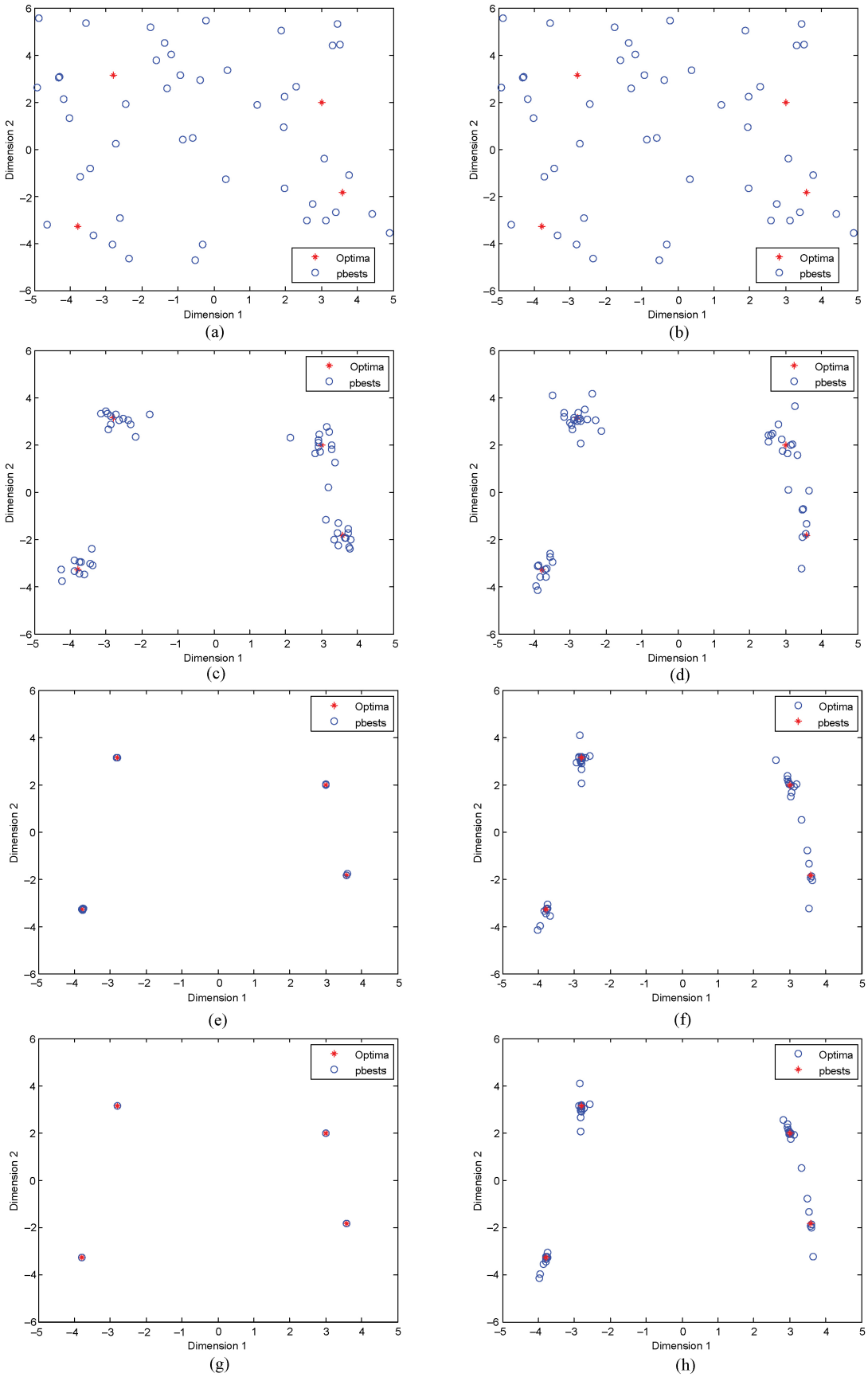

Fig. 4. Distributions of pbests of LIPS and $r2$pso on different stages (F5). (a) Iteration 1 (LIPS). (b) Iteration 1 ($r2$pso). (c) Iteration 20 (LIPS). (d) Iteration 20 ($r2$pso). (e) Iteration 100 (LIPS). (f) Iteration 100 ($r2$pso). (g) Iteration 200 (LIPS). (h) Iteration 200 ($r2$pso).

once the particles are crowded around different niches, LIPS performs the search within each niche independently. This greatly improves the fine-tuning ability of LIPS. In order to determine the statistical significance of the advantage of LIPS, *t*-test (all compared with LIPS) is applied and shown in the last row of each test function. The numerical values 1, 0 represent that other methods are statistically inferior to or equal to the proposed algorithm. The last three rows of both tables summarize how many cases that LIPS perform better, similar, or worse than other algorithms. From the results, we can observe that LIPS always perform better or equal when compared with other niching methods on all test functions.

C. Performance on High-Dimensional Problems (20-D and 30-D)

To test the performance of proposed algorithm on high-dimensional problems, 20-D and 30-D composition functions (F16–F20) are used. All the experimental settings are the same as 10-D composition functions. The results are presented in Table IV. Unlike FERPSO, SDE, and CDE, the performances of LIPS and SACMA-ES only drop slightly when the dimension increases. LIPS still performs the best out of all 10 compared algorithms. From these results, we can conclude that LIPS is able to generate stable performance over a wide variety of test functions.

D. Advantage of Distance-Based Neighborhood

To show the advantage of distance-based neighborhood over topological neighborhood, LIPS is compared with *r*2pso on F5 Himmelblau's function. The distributions of pbests of both algorithms in different iterations are plotted in Fig. 4. As we can see from the plots, LIPS converges much faster than *r*2pso, especially at later stage. For *r*2pso, the distributions of pbests from iteration 100 to iteration 200 do not change much, which means that many function evaluations are wasted due to the problem mentioned in Section III-B. In contrast, LIPS uses the information of the closest neighbors that are likely to target on the same peak. This search mechanism is able to increase the convergence speed and accuracy simultaneously.

E. Effect of Neighborhood Size

The performance of the proposed LIPS depends on the neighborhood size. A smaller neighborhood size will generate a better diversity for the population while a larger neighborhood size is good for convergence. Table V shows the performances on test F11 and F16 with the neighborhood varied from 2 to 5. For test F11, the average number of optima obtained drops as the neighborhood size increases. This is because the number of global optima for F11 is high. Although a large neighborhood size increases the convergence, it decreases the diversity and misses some of the peaks. On the other hand, F16 is composite function and finding one peak is already challenging. Therefore, the convergence is more important than diversity. How to choose the neighborhood size depends on the users need. If the target is diversity, a smaller neighborhood size should be used or otherwise a larger neighborhood size should be selected. In this paper, a

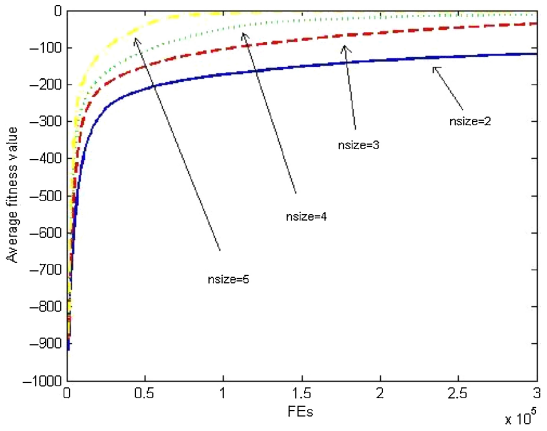

Fig. 5. Convergence graph for different neighborhood size (F16).

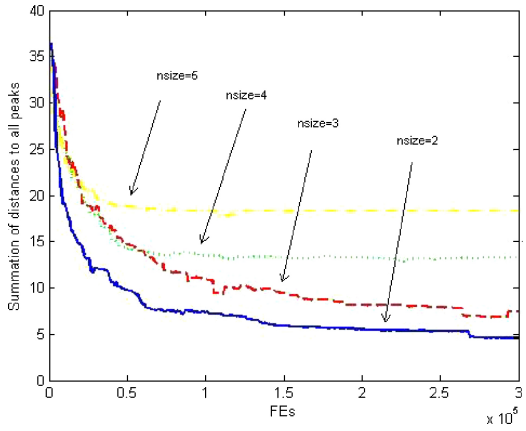

Fig. 6. Summation of distance to optima (F16).

dynamic neighborhood size is used. The neighborhood size is linearly increased from 2 to 5 over the function evaluations. In order to show the effect of neighborhood size more clearly, the convergence graph and distance to optima over function evaluations on F16 are shown in Figs. 5 and 6. From the convergence graph we can see that a larger neighborhood size generates a higher average fitness value of the population while a smaller neighborhood size generates small summation of distance to all peaks.

F. Maintaining Optima Once Found

As stated by Mahoud [3], a good niching algorithm must be able to locate global optima and maintain them for an exponential to infinite time period, with respect to population size. Maintaining found optima is important for an effective and stable multimodal optimization algorithm. LIPS is able to find and maintain the found optima until the end of the run. This is because the proposed algorithm uses neighborhood information to execute local search. Once a niche is formed around one global peak, the algorithm will continue to search better solutions within the same niche. Only when a better solution is found within the niche, it replaces the current personal best that can maintain the found peak until the end of the run. Fig. 7 shows the niching behavior of LIPS on F1. From the figure we can see that LIPS is able to develop stable niches around the global peaks.

TABLE VI
PEAK RATIO ON HIGH-DIMENSIONAL PROBLEMS (F16−F20)

Test Function		LIPS	*r*2pso	*r*3pso	*r*2pso-*lhc*	*r*3pso-*lhc*	SPSO	FERPSO	SDE	CDE	SACMA-ES
F16 (20-D)	Worst	0.38	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.25
	Best	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.25
	Mean	**0.46**	0.00	0.00	0.00	0.00	0.00	0.00	0.13	0.00	0.25
	Std	0.07	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	1
F17 (20-D)	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.33
	Best	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33
	Mean	**0.33**	0.00	0.00	0.00	0.00	0.00	0.00	0.22	0.00	**0.33**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.09	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	0
F18 (20-D)	Worst	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.33
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.50
	Mean	**0.65**	0.00	0.00	0.00	0.00	0.00	0.00	0.18	0.00	0.45
	Std	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.00	0.08
	t-test	–	1	1	1	1	1	1	1	1	1
F19 (20-D)	Worst	0.17	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Mean	**0.32**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Std	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	1
F20 (20-D)	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.17	0.00	0.17
	Best	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.33	0.00	0.33
	Mean	**0.33**	0.00	0.00	0.00	0.00	0.00	0.00	0.20	0.00	0.32
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.07	0.00	0.05
	t-test	–	1	1	1	1	1	1	1	1	0
F16 (30-D)	Worst	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.25
	Best	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.38
	Mean	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.26**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.04
	t-test	–	1	1	1	1	1	1	1	1	0
F17 (30-D)	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	Best	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	Mean	**0.33**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.33**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	0
F18 (30-D)	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	Best	0.67	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.50
	Mean	**0.50**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.35
	Std	0.12	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.05
	t-test	–	1	1	1	1	1	1	1	1	1
F19 (30-D)	Worst	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Best	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Mean	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	0	0	0	0	0	0	0	0	0
F20 (30-D)	Worst	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	Best	0.33	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.33
	Mean	**0.33**	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.33**
	Std	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	t-test	–	1	1	1	1	1	1	1	1	0
t-test summary	Better	–	9	9	9	9	9	9	9	9	4
	Similar	–	1	1	1	1	1	1	1	1	6
	Worst	–	0	0	0	0	0	0	0	0	0

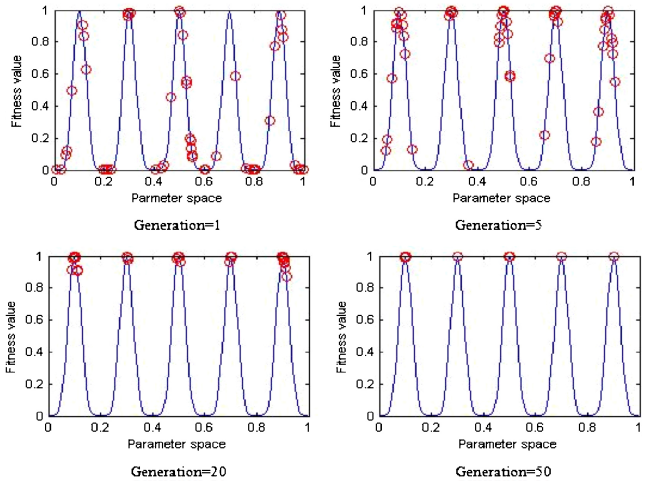

Fig. 7. Distribution of population over function evaluations (F1).

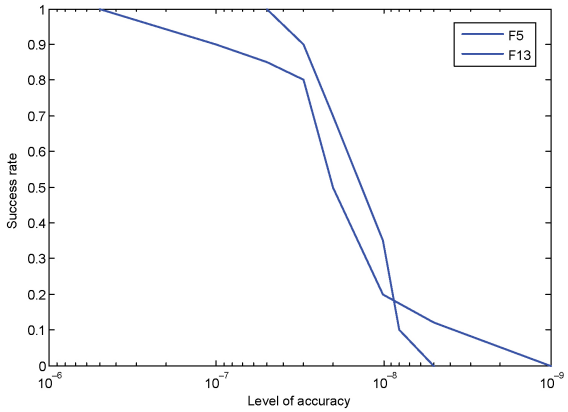

Fig. 8. Effect of varying level of accuracy.

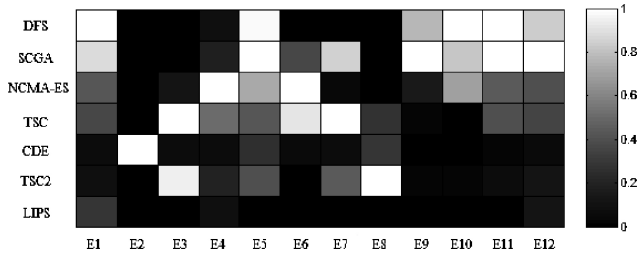

Fig. 9. Overview of peak accuracies of each algorithm. Results are normalized for each problem, so that 0 refers to the best and 1 refers to the worst algorithm.

G. Effect of Varying Level of Accuracy

The parameter ε (level of accuracy) can affect the results of the algorithm greatly. To demonstrate the effect of varying level of accuracy, the plot (Fig. 8) on level of accuracy versus success rate is generated based on test function F5 (Himmelblau's function) and F13 (Michalewicz function). From the plot we can observe that the success rate drops when the level of accuracy is increased.

H. Comparison With Results Reported in [47]

In order to further demonstrate the superior performance of LIPS, it was compared with the results reported in one of the latest multimodal optimization works [47]. The test function, parameter settings, and results of the other compared algorithms are directly taken from [47]. The assessment criteria are also adopted from [47], which are listed as follows.

TABLE VII
PEAK RATIO FOR FUNCTIONS F11 AND F16 ILLUSTRATING THE EFFECT OF NEIGHBORHOOD SIZE

Test Function		nsize=2	nsize=3	nsize=4	nsize=5
F11	Worst	0.94	0.89	0.83	0.72
	Best	1.00	1.00	1.00	1.00
	Mean	**0.99**	0.96	0.93	0.88
	Std	0.02	0.03	0.06	0.07
F16	Worst	0.00	0.25	0.50	0.50
	Best	0.13	0.50	0.63	0.63
	Mean	0.03	0.38	0.54	**0.59**
	Std	0.05	0.08	0.07	0.06

1) *Peak accuracy*: For each desired peak to be located, the closest individual X in the population is taken and absolute difference in objective values is calculated. If the objective value of individual X is denoted by $f(X)$, the peak accuracy is calculated using the following equation:

$$\text{peak accuracy} = \sum_{i=1}^{\#\text{ peaks}} \left| f(\text{peak}_i) - f(X) \right|.$$

2) *Distance accuracy*: Peak accuracy may lead to erroneous results, if the peaks are close to each other or with identical height. The distance accuracy is used to avoid this error. It is calculated the same way as peak accuracy, with the only change that the fitness values are replaced by the Euclidean distance as follows:

$$\text{distance accuracy} = \sum_{i=1}^{\#\text{ peaks}} (\text{Euclidean distance between peak}_i \text{ and individual } X).$$

The performance results are shown in Tables VIII and IX. Fig. 9 enables a clear visual comparison for the peak accuracies of every function over all algorithms. From the results we can observe that LIPS performs the best out of the seven compared algorithms, especially on test functions E8–E11. The ranks of each algorithm are shown in the bracket while the results for the best algorithm are highlighted in boldface. For test function E7, dynamic fitness sharing (DFS) algorithm ranks the best in terms of peak accuracy and ranks the last in terms of distance accuracy. This is due to the peak accuracy error mentioned above, as DFS totally ignore some of the peaks for E7. Generally, distance accuracy is able to effectively reflect the performance of each algorithm. The superior performance of LIPS is again due to the same reason (high convergence speed and high fine search accuracy).

VI. CONCLUSION

In this paper, a LIPS niching algorithm was proposed to solve multimodal problems. LIPS made use of the neighborhood information to realize niching behavior. LIPS also

TABLE VIII
RESULTS OF PEAK ACCURACY

Test Function	LIPS	TSC2	CDE	TSC	NCMA-ES	SCGA	DFS
E1: Waves	6.18 (3)	1.84 (2)	**1.59 (1)**	7.7 (4)	8.89 (5)	18.59 (6)	20.93 (7)
E2: Sphere	**2.31E-42 (1)**	1.81E-07 (2)	4.48E-04 (7)	4.90E-06 (6)	3.92E-06 (4)	2.86E-07 (3)	4.17E-06 (5)
E3: Shifted Rastrigin	**0 (1)**	1.63 (6)	0.11 (4)	1.73 (7)	0.19 (5)	0.01 (2)	0.02 (3)
E4: Hybrid composition	163.46 (2)	369.93 (5)	**134.64 (1)**	934.45 (6)	1840 (7)	317.24 (4)	164.85 (3)
E5: Rescaled six-hump	**1.97E-05 (1)**	2.77 (3)	1.78 (2)	2.99 (4)	5.12 (5)	7.06 (7)	6.89 (6)
E6: Branin RCOS	**1.52E-05 (1)**	0.02 (3)	0.1 (4)	1.79 (6)	1.96 (7)	0.73 (5)	3.42E-04 (2)
E7: Shubert	9.90 (2)	727.9 (5)	115.4 (4)	1628.46 (7)	52.6 (3)	1381.05 (6)	**0.11 (1)**
E8: Ackely	**3.23E-11 (1)**	0.85 (7)	0.24 (6)	0.23 (5)	0.01 (4)	0.003 (2.5)	0.003 (2.5)
E9: Michalewicz	**4.39E-05 (1)**	0.009 (3)	0.006 (2)	0.01 (4)	0.07 (5)	0.48 (7)	0.37 (6)
E10: Ursem F1	**3.18E-05 (1)**	0.1 (4)	0.002 (2)	0.005 (3)	0.64 (5)	0.76 (6)	0.92 (7)
E11: Ursem F3	**8.85E-05 (1)**	0.32 (3)	0.1 (2)	1.68 (4)	1.89 (5)	4.28 (6)	4.3 (7)
E12: Ursem F4	0.28 (2.5)	0.28 (2.5)	**0.12 (1)**	0.89 (4)	1.02 (5)	2.53 (7)	2.11 (6)
Total rank	**17.5**	45.5	36	60	60	61.5	55.5

TABLE IX
RESULTS OF DISTANCE ACCURACY

Test Function	LIPS	TSC2	CDE	TSC	NCMA-ES	SCGA	DFS
E1: Waves	1.01 (3)	0.79 (2)	**0.41 (1)**	3.26 (4)	3.87 (5)	11.56 (7)	11.52 (6)
E2: Sphere	**9.40E-22 (1)**	9.32E-05 (3)	5.25E-03 (7)	9.08E-05 (2)	5.84E-04 (5)	1.65E-04 (4)	8.12E-04 (6)
E3: Shifted Rastrigin	**0 (1)**	0.05 (5)	0.03 (4)	0.07 (6)	0.14 (7)	0.01 (2)	0.02 (3)
E4: Hybrid composition	0.16 (2)	0.49 (3)	**0.07 (1)**	1.07 (5)	0.71 (4)	2.51 (6)	3.05 (7)
E5: Rescaled six-hump	**5.72E-05 (1)**	0.54 (3)	0.22 (2)	3.94 (5)	1.48 (4)	5.31 (7)	4.55 (6)
E6: Branin RCOS	**2.50E-03 (1)**	0.45 (3)	0.21 (2)	5.48 (6)	4.56 (5)	6.04 (7)	3.63 (4)
E7: Shubert	**0.12 (1)**	33.2 (5)	3.12 (2)	59.2 (6)	31.0 (4)	22.1 (3)	88.2 (7)
E8: Ackely	**1.14E-11 (1)**	0.21 (7)	0.05 (5)	0.06 (6)	2.96E-03 (4)	9.78E-04 (3)	8.18E-04 (2)
E9: Michalewicz	**0 (1)**	0.02 (3)	0.01 (2)	0.03 (4)	0.15 (5)	0.92 (7)	0.72 (6)
E10: Ursem F1	**1.32E-07 (1)**	0.21 (4)	9.00E-03 (2)	0.01 (3)	1.42 (5)	1.55 (6)	1.87 (7)
E11: Ursem F3	**2.10E-05 (1)**	0.34 (3)	0.12 (2)	1.77 (4)	1.99 (5)	4.32 (7)	4.29 (6)
E12: Ursem F4	0.83 (2)	0.95 (3)	**0.36 (1)**	7 (5)	5.42 (4)	7.68 (7)	7.36 (6)
Total rank	**16**	44	31	57	57	66	66

eliminated the need to specify any major niching parameters. With the neighborhood information, the algorithm was enhanced with local search ability. Experimental results also showed that the proposed LIPS algorithm can outperform several well-known niching algorithms compared in this paper on 30 standard benchmark functions in a statistically meaningful way.

A first step toward extending this paper would be to add certain diversity enhancement techniques, as this will surely increase the probability of more global or local peaks for complex multimodal optimization problems. Secondly, it would be interesting to self-adapt neighborhood size and population size, as this might further improve the performance of the current algorithm. Also, the proposed approach could be applied to solving multimodal optimization in a dynamic environment.

REFERENCES

[1] T. Bäck, D. Fogel, and Z. Michalewicz, *Handbook of Evolutionary Computation*. Oxford, U.K.: Oxford Univ. Press, 1997.

[2] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. New York: Springer, 2003.

[3] S. W. Mahfoud, "Niching methods for genetic algorithms," Ph.D. dissertation, Dept. Comput. Sci., Univ. Illinois Urbana-Champaign, Urbana, 1995.

[4] S. W. Mahfoud, "A comparison of parallel and sequential niching methods," in *Proc. 6th Int. Conf. Genet. Algorithms*, Jul. 1995, pp. 136–143.

[5] J. E. Vitela and O. Castanos, "A real-coded niching memtic algorithm for continuous multimodal function optimization," in *Proc. IEEE Congr. Evol. Computat.*, Jun. 2008, pp. 2170–2177.

[6] O. Mengsheol and D. Goldberg, "Probabilistic crowding: Deterministic crowding with probabilistic replacement," in *Proc. GECCO*, Jul. 1999, pp. 409–416.

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

[8] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *Proc. IEEE Congr. Evol. Computat.*, Jun. 2004, pp. 1382–1389.

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

[10] G. R. Harik, "Finding multimodal solutions using restricted tournament selection," in *Proc. 6th Int. Conf. Genet. Algorithms*, 1995, pp. 24–31.

[11] A. Petrowski, "A clearing procedure as a niching method for genetic algorithms," in *Proc. 3rd IEEE Congr. Evol. Computat.*, May 1996, pp. 798–803.

[12] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micromach. Human Sci.*, vol. 1. Mar. 1995, pp. 39–43.

[13] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, Nov.–Dec. 1995, pp. 1942–1948.

[14] J. Kennedy, R. C. Eberhart, and Y. Shi, *Swarm Intelligence*. San Francisco, CA: Morgan Kaufmann, 2001.

[15] Y. Shi and R. C. Eberhart, "Monitoring of particle swarm optimization," *Frontiers Comput. Sci. China*, vol. 3, no. 1, pp. 31–37, 2009.

[16] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.
[17] L. Liu, S. Yang, and D. Wang, "Particle swarm optimization with composite particles in dynamic environments," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 40, no. 6, pp. 1634–1648, Dec. 2010.
[18] M. Daneshyari and G. G. Yen, "Culture-based multiobjective particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 41, no. 2, pp. 553–567, Apr. 2011.
[19] S. Kiranyaz, T. Ince, A. Yildirim, and M. Gabbouj, "Fractional particle swarm optimization in multidimensional search space," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 40, no. 2, pp. 298–319, Apr. 2010.
[20] R. A. Krohling and L. S. Coelho, "Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems," *IEEE Trans. Syst., Man, Cybern., Part B*, vol. 36, no. 6, pp. 1407–1416, Dec. 2006.
[21] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ration," in *Proc. Genet. Evol. Computat. Conf.*, 2007, pp. 78–85.
[22] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genet. Evol. Computat. Conf.*, vol. 3102. 2004, pp. 105–116.
[23] X. Li, "Niching without niching parameters: Particle swarm optimization using a ring topology," *IEEE Trans. Evol. Computat.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.
[24] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahitis, "Stretching technique for obtaining global minimizers through particle swarm optimization," in *Proc. Particle Swarm Optimiz. Workshop*, 2001, pp. 22–29.
[25] D. Beasley, D. R. Bull, and R. R. Martin, "A sequential niche technique for multimodal function optimization," *Evol. Comput.*, vol. 1, no. 2, pp. 101–125, Jun. 1993.
[26] K. Parsopoulos and M. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Computat.*, vol. 8, no. 3, pp. 211–224, Jun. 2004.
[27] K. Parsopoulos and M. Vrahatis, "Modification of the particle swarm optimizer for locating all the global minima," in *Artificial Neural Networks and Genetic Algorithms*. New York: Springer, 2001, pp. 324–327.
[28] R. Brits, A. Engelbrecht, and F. van den Bergh, "A niching particle swarm optimizer," in *Proc. 4th Asia-Pacific Conf. SEAL*, 2002, pp. 692–696.
[29] Y. Shi and R. C. Eberhart, "Population diversity of particle swarms," in *Proc. IEEE Congr. Evol. Computat.*, Jun. 2008, pp. 1063–1067.
[30] N. Q. Uy, N. X. Hoai, R. I. Maccay, and P. M. Tuan, "Initializing PSO with randomized low-discrepancy sequences: The comparative results," in *Proc. Congr. Evol. Computat.*, 2007, pp. 1985–1992.
[31] J. Kennedy, "The particle swarm: Social adaptation of knowledge," in *Proc. Int. Conf. Evol. Computat.*, 1997, pp. 303–308.
[32] T. Hendtlass, "Particle swarm optimisation and high dimensional problem spaces," in *Proc. IEEE Congr. Evol. Computat.*, May 2009, pp. 1988–1994.
[33] R. Brits, A. P. Engelbrecht, and F. van den Bergh, "Solving systems of unconstrained equations using particle swarm optimizers," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2002, pp. 102–107.
[34] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. 7th Annu. Conf. Evol. Program.*, Mar. 1998, pp. 591–600.
[35] 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.
[36] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genet. Evol. Computat.*, 2005, pp. 873–880.
[37] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Computat.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.
[38] H. G. Beyer and H. P. Schwefel, "Evolution strategies: A comprehensive introduction," *Nat. Comput.: An Int. J.*, vol. 1, no. 1, pp. 3–52, 2002.
[39] O. M. Shir, M. Emmerich, and T. Back, "Adaptive niche radii and niche shapes approaches for niching with the CMA-ES," *Evol. Computat.*, vol. 18, no. 1, pp. 97–126, 2010.
[40] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: Simpler, maybe better," *IEEE Trans. Evol. Computat.*, vol. 8, no. 3, pp. 204–210, Jun. 2004.
[41] B. Y. Qu and P. N. Suganthan, "Novel multimodal problems and differential evolution with ensemble of restricted tournament selection," in *Proc. IEEE Congr. Evol. Computat.*, Jul. 2010, pp. 3480–3486.
[42] K. Deb, "Genetic algorithms in multimodal function optimization, the Clearinghouse for Genetic Algorithms," M.S. thesis, Dept. Eng. Mechanics, Univ. Alabama, Tuscaloosa, TCGA Rep. 89002, 1989.
[43] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. New York: Springer-Verlag, 1996.
[44] K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, Dept. Comput. Commun. Sci., Univ. Michigan, Ann Arbor, 1975.
[45] R. K. Ursem, "Multimodal evolutionary algorithms," in *Proc. Congr. Evol. Comput.*, vol. 3. 1999, pp. 1633–1640.
[46] J. Gan and K. Warwick, "A variable radius niching technique for speciation in genetic algorithms," in *Proc. GECCO*, 2000, pp. 96–103.
[47] C. Stoean, M. Preuss, R. Stoean, and D. Dumitrescu, "Multimodal optimization by means of a topological species conservation algorithm," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 842–864, Dec. 2010.
[48] T. Hendtlass, "The particle swarm algorithm," in *Computational Intelligence: A Compendium*. New York: Springer, 2008, pp. 1029–1062.
[49] T. Hendtlass, "Fitness estimation and the particle swarm optimisation algorithm," in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 4266–4272.

B. Y. Qu received the B.E. and Ph.D. degrees from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, in 2007 and 2012, respectively.

He is currently a Research Staff Member and a Lecturer with the Zhongyuan University of Technology, Zhengzhou, China. His current research interests include machine learning, neural networks, genetic and evolutionary algorithms, swarm intelligence, and multiobjective optimization.

Ponnuthurai Nagaratnam Suganthan (S'91–M'92–SM'00) received the B.A., Postgraduate Certificate, and M.A. degrees in electrical and information engineering from the University of Cambridge, Cambridge, U.K., in 1990, 1992, and 1994, respectively, and the Ph.D. degree from the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

From 1999 to 2003, he was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, where he is currently an Associate Professor. His current research interests include evolutionary computation, pattern recognition, multiobjective evolutionary algorithms, bioinformatics, applications of evolutionary computation, and neural networks.

Dr. Suganthan received the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award in 2012 for his article "Self-adaptive differential evolution (SaDE)" published in April 2009. He is an Editorial Board Member of the *Evolutionary Computation Journal* (MIT Press). He is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Information Sciences* (Elsevier), *Pattern Recognition* (Elsevier), and the *International Journal of Swarm Intelligence Research*. He was a founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation*, an Elsevier Journal.

Swagatam Das (M'10) received the B.E.Tel.E., M.E.Tel.E. (control engineering specialization), and Ph.D. degrees from Jadavpur University, Kolkata, India, in 2003, 2005, and 2009, respectively.

From 2006 to 2011, he was an Assistant Professor with the Department of Electronics and Telecommunication Engineering, Jadavpur University. Currently, he is a Visiting Assistant Professor with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata. He has published more than 130 research articles in peer-reviewed journals and international conferences. His current research interests include evolutionary computing, pattern recognition, multiagent systems, and wireless communication.

Dr. Das is the founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation*, an international journal from Elsevier. He is an Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS (PART-A) and *Information Sciences* (Elsevier). He is an Editorial Board Member of the *International Journal of Artificial Intelligence and Soft Computing* and the *International Journal of Adaptive and Autonomous Communication Systems*.