A Generic Archive Technique for Enhancing the Niching Performance of Evolutionary Computation

Yu-Hui Zhang, Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, and Jun Zhang*
Department of Computer Science, Sun Yat-sen University
Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education
Engineering Research Center of Supercomputing Engineering Software, Ministry of Education
School of Advanced Computing, Sun Yat-sen University, Guangzhou, China
*Corresponding Author: junzhang@ieee.org

Abstract—**The performance of a multimodal evolutionary algorithm is highly sensitive to the setting of population size. This paper introduces a generic archive technique to reduce the importance of properly setting the population size parameter. The proposed archive technique contains two components: subpopulation identification and convergence detection. The first component is used to identify subpopulations in a number of individuals while the second one is used to determine whether a subpopulation is converged. By using the two components, converged subpopulations are identified, and then, individuals in the converged subpopulations are stored in an external archive and re-initialized to search for other optima. We integrate the archive technique with several state-of-the-art PSO-based multimodal algorithms. Experiments are carried out on a recently proposed multimodal problem set to investigate the effect of the archive technique. The experimental results show that the proposed method can reduce the influence of the population size parameter and improve the performance of multimodal algorithms.**

Keywords—archive; multimodal optimization; niching technique; Particle Swarm Optimization

I. Introduction

Many real-world problems have multiple satisfactory solutions. Multimodal optimization aims to locate all these solutions simultaneously. Tracing several high-quality solutions has two advantages. First, the distribution of good solutions can provide useful information about the problem domain. Second, when it happens that an optimal solution is difficult to achieve, we can quickly switch to another solution without incurring significant performance loss. To this end, many efficient multimodal optimization algorithms have been proposed [1].

Evolutionary algorithms (EAs), which are originally designed to locate a single global optimum, have been adapted to tackle multimodal optimization problems. Generally, there are two kinds of strategies of applying EAs to multimodal problems: (1) locate one optimum at a time and (2) locate all optima at one time. The first strategy attempts to handle the task of locating multiple optima by running an EA several times, each time hoping to find a different optimum. However, the probability of an optimum being found is closely related to the shape and size of its basin of attraction. Individuals of an EA are more likely to converge to the basin of attraction with the largest size, resulting in that the outputs of every run are mostly the same. Note that EAs are population-based methods with natural parallel search ability, they can be used to simultaneously locate multiple optima by incorporating a technique called "niching". Niching [2] is a generic term refers to the technique of finding and preserving multiple niches, with the aim of preventing the population converge to a single optimum. Many niching techniques have been proposed in the literature, including fitness sharing [3], restricted tournament selection [4], crowding [5], speciation [6], and clearing [7], etc.

The latter strategy (parallel niching strategy) agrees well with the parallel search nature of EAs and has exhibited better performance over the former (sequential niching strategy) [8]. Using a parallel niching strategy, the whole population of an EA is divided into several subpopulations and these subpopulations are maintained during the evolutionary process, so as to simultaneously locate multiple optima. Each subpopulation is responsible for searching one of the optima. It can be inferred that the required population size of an EA depends largely on the number of optima in the problem landscape. However, as the number of optima for a given problem is generally unknown, it remains a difficult task to determine the suitable population size. Moreover, the convergence speed of each subpopulation may differ from one another. It is a waste of computational resources if subpopulations that have already converged are kept evolving.

To alleviate the above problems, in this paper, we propose a third niching strategy that locates between the traditional sequential and parallel niching. This strategy is developed by relaxing the restriction of parallel niching, i.e., subpopulations search for different optima in parallel, but individuals in the converged subpopulations are reinitialized to search other promising areas instead of wandering around a single optimum. The re-initialization mechanism makes the search behavior of each subpopulation similar to that of the sequential niching strategy somewhat. Compared with the parallel niching strategy, the proposed method is more efficient since no computational efforts are spent on converged subpopulations. Further, this strategy reduces the influence of the population size parameter, for individuals can be reused to locate different optima.

This work was supported in part by the National High-Technology Research and Development Program (863 Program) of China No.2013AA01A212, in part by the NSFC for Distinguished Young Scholars 61125205, in part by the NSFC No. 61332002 and No.61300044.

Before the re-initialization of a converged subpopulation, it is necessary to store the best individual in the subpopulation in an external archive to avoid losing an optimum. However, deciding when to archive the individual is not a simple task. This paper handles the task by introducing a generic archive technique. The archive technique contains two components. The first component is designed to identify the subpopulations in a number of individuals and the second is used to determine whether a subpopulation is converged. By combining these two components we can identify converged subpopulations. We then add the individuals in the converged subpopulations to an archive and reinitialize the archived individuals.

Particle swarm optimization (PSO) [9]-[11] is among the most popular population-based search algorithms in the evolutionary computation community. It is conceptually simple and has shown to be very effective in solving optimization problems. In the literature, PSO has been adopted to solve multimodal problems [21]-[23]. To evaluate the effectiveness of the proposed archive technique, several PSO-based multimodal algorithms are integrated with the archive technique and are tested on the CEC2013 benchmark set [12]. The capability of reducing the influence of population size is demonstrated by the comparison between the archive-integrated algorithms and the original multimodal algorithms.

The rest of this paper is organized as follows. Section II briefly describes the PSO algorithm and reviews some niching PSO algorithms used for multimodal optimization. Section III introduces our archive technique in sufficient detail and shows how to integrate the archive technique with the PSO-based multimodal algorithms. Experimental results on CEC2013 benchmarks are presented and discussed in Section IV. Section V concludes this paper.

II. Particle Swarm Optimization For Multimodal Optimization

PSO is one of the most promising population-based search algorithm for solving optimization problems. In the literature, PSO has also been adopted to solve multimodal problems. Parsopoulos and Vrahatis [13][14] introduced an approach which isolates particles whose accuracy have reached a specific level and then uses a “stretching” technique to repel other particles from moving towards them. A small population of particles is generated around the isolated particles to perform a finer search. Brits *et al.* [15] proposed *nbest* PSO which utilizes neighborhood information of particles. Instead of using gbest, each particle is guided by the average of the positions of its *k*-nearest neighbors. In the same year, Brits *et al.* [16] introduced a PSO variant called NichePSO. In NichePSO, a subpopulation is created if a particle’s fitness value has been unchanged for a small number of iterations. Multiple subpopulation are generated in this way. These subpopulation can merge together or absorb particles from the main swarm. Li [17] employed the concept of speciation to PSO and proposed speciation-based PSO (SPSO). In SPSO, particles are divided into species according to their similarity. Each species is grouped around a dominated particle called species seed. At each iteration, species seeds are first identified. Particles fall within the radius of a species seed are assigned to that species. Instead of using gbest, every particle uses its own species seed to update its velocity. Schoeman and Engelbrecht [18] proposed a parallel vector-based PSO (PVPSO) which uses a set of vector operations to form niches in the search space. Özcan and YÕlmaz [19] proposed a PSO algorithm with craziness and hill climbing (CPSO). In CPSO, the main swarm is divided into sub-swarms of size *n* according to the particles’ geographical positions. A random walk component and a hill climber are utilized to enhance PSO’s exploration and exploitation capabilities. Barrera and Coello [20] give a review of PSO-based multimodal algorithms. In the following paragraphs, we briefly discuss several niching PSOs that have been proposed more recently.

1) Fitness Euclidean-Distance Ratio PSO (FERPSO) : Most of existing multimodal evolutionary algorithms introduce one or more control parameters, which are very difficult to set without prior knowledge of a problem. To remove the need of these niching parameters, Li [21] proposed a PSO based on fitness Euclidean-distance Ratio (FER). In FERPSO, each particle moves towards its pbest and best neighbor. The best neighbor of a particle is chosen according to FER. For the *i*-th particle, its FER value in connection with the *j*-th particle is computed as follows:

$$FER_{(j,i)} = \alpha \cdot \frac{f(\text{pbest}_j) - f(\text{pbest}_i)}{\| \text{pbest}_j - \text{pbest}_i \|} \tag{1}$$

where α is a scaling factor, $f(x)$ is the fitness function, $\|\text{pbest}_j-\text{pbest}_i\|$ denotes the Euclidean-distance between pbest_j and pbest_i. The particle that used to obtain the highest FER value is identified as the best neighbor of the *i*-th particle. By using nbest instead of gbest, multiple niches are naturally formed around multiple optima.

2) Ring Topology PSO (r2pso, r3pso): With the same purpose of FERPSO, PSO using a ring topology (rpso) has also been recommended to tackle multimodal problems [22]. rpso does not require any niching parameters and is very simple. In rpso, particles are arranged in a circle and each particle only interacts with its direct neighbors. r2pso and r3pso are two versions of rpso. In r2pso, each particle interacts only with its immediate neighbor on its left, whilst in r3pso, each particle interacts with both its left and right neighbors. The ring topology PSO is found to be able to form multiple stable niches. Experimental results reported in [22] show that PSO using ring topology is very promising in solving multiobjective problems.

3) Distance-Based Locally Informed Particle Swarm (LIPS): To avoid all particles converge to a single optimum, Qu *et al.* [23] presented a distance-based locally informed particle swarm (LIPS) optimizer. LIPS eliminates the need for niching parameters and enhance the fine search ability of PSO. The search behavior of a particle is guided by the local information of its nearest neighbors. In LIPS, the velocity update of the *i*-th particle is changed to:

$$V_i^d = \omega \cdot (V_i^d + \varphi(P_i^d - X_i^d)) \quad (2)$$

where

$$P_i = \frac{\sum_{j=1}^{nsize} (\varphi_j \cdot \text{nbest}_j)}{\varphi}. \quad (3)$$

φ_j is a uniformly distributed random number in the range of [0, 4.1/*nsize*] and φ is the sum of all φ_j. nbest_j represents the *j*-th nearest neighbor of pbest_i. *nsize* is the neighborhood size and is dynamically increased from 2 to 5 over the function evaluations.

The majority of the aforementioned multimodal algorithms attempt to locate all the optima at one time. Therefore, the population size must be set according to the number of optima of the problem to be solved. To alleviate the need to choose the suitable population size via trial-and-error, this paper suggests to use the third strategy and proposes an archive technique. The detailed description of the proposed archive technique is presented in the following section.

III. The Proposed Archive Technique

With the aim of reducing the influence of the population size parameter and improving the performance of a PSO-based multimodal algorithm, we reinitialize individuals in converged subpopulations to search for other optima. Before re-initializing of a subpopulation, the best individual in the subpopulation is stored in an external archive to avoid losing a possible optimum. To identify converged subpopulations, there are two problems to be addressed, namely, (1) How to find out the subpopulations in a number of individuals? (2) How to determine whether a subpopulation is converged? The proposed archive technique is intended to solve the two problems. It has two components, subpopulation identification and convergence detection, which are detailed in the following subsections.

A. Subpopulation Identification

Mahoud [2] stated that a good niching technique must be able to locate global optima and maintain them for an exponential to infinite time period. Many niching techniques described in the previous section are qualified with respect to the statement. By using these niching techniques, multiple niches are formed around optima. The subpopulation identification is used to find these niches out when given a number of individuals. The well-known *k*-means algorithm is not suitable here, for its input parameter *k* is among the objectives of the identification task (*k*-means is a well-known clustering algorithm, where *k* is a user-specific parameter that indicates the number of clusters [24]). In this paper, we design an alternative algorithm that does not require prior knowledge of the number of subpopulations. The detailed procedure is listed in Algorithm 1.

Fig. 1 gives an example to show how the algorithm works. In Fig. 1, there are two subpopulations that contain five and six individuals respectively. For each individual, we find its *k* nearest neighbors (in this example, *k*=3). The one-way arrow from one node to another indicates that the latter node is one of the *k*-nearest neighbors of the former. The double arrow

Algorithm 1 Subpopulation Identification

```
Step 1. Compute the distance matrix $dis[NP][NP]$ of the population //$dis[i][j]$ denotes the Euclidean distance between the $i$-th individual and the $j$-th individual.
Step 2. For each individual, find its $k$ nearest neighbors according to the distance matrix.
Step 3. Construct a neighborhood matrix Nei[$NP$][$NP$]. Nei[$i$][$j$] is set to 1 if the $j$-th individual is one of the $k$ nearest neighbors of the $i$-th individual, otherwise, Nei[$i$][$j$] is set to 0.
    Repeat
Step 4. Join pairs of individuals that are mutual neighbors. Individual $i$ and individual $j$ are mutual neighbors if Nei[$i$][$j$]==1 and Nei[$j$][$i$]==1.
Step 5. Modify the neighbor matrix. Suppose that the joint individual of $i$ and $j$ is represented by $u$, for another individual $v$ in the population, Nei[$u$][$v$]=1 if and only if Nei[$i$][$v$]=1 or Nei[$j$][$v$]=1, Nei[$v$][$u$]=1 if and only if Nei[$v$][$i$]=1 or Nei[$v$][$j$]=1
    Until there are no mutual neighbors
```

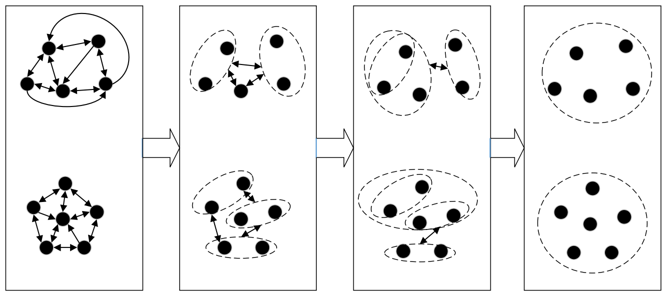


Fig. 1. Illustration of subpopulation identification.

connector indicates that two nodes are among the nearest neighbors of each other, which are named mutual neighbors. We join pairs of mutual neighbors and modify the arrow connectors as described in Step 4 and Step 5. Then, the above process is repeated until there are no mutual neighbors. Finally, we get the two subpopulations.

In the proposed subpopulation identification, *k* determines the out-degree of each node in the mutual *k*-nearest neighbor graph. It is a problem-independent control parameter. Nevertheless, we find that *k*=5 to 7 is sufficient to join individuals in the same subpopulation and to separate different subpopulations.

B. Convergence Detection

After subpopulations have been identified, we use two criteria to determine whether a subpopulation is converged. Before introducing the criteria, we first define a quantity called convergence factor (CVF) as follows:

$$CVF = \sum_{i=1}^{nsize} \min\{\| X_i - X_j \|, 1 \le j \le nsize \wedge j \ne i\} \Big/ nsize \quad (4)$$

where *nsize* is the size of the subpopulation, $\|X_i - X_j\|$ is the Euclidean distance between X_i and X_j. We use $mindis_i$ to denote the distance between individual *i* and its nearest neighbor. CVF is the average of $mindis_i$, *i*=1, 2, …, *nsize*. It can be used as an indicator to show how close individuals are to one another. Intuitively, successive decreases in CVF mean that the subpopulation is converging to a possible optimum.

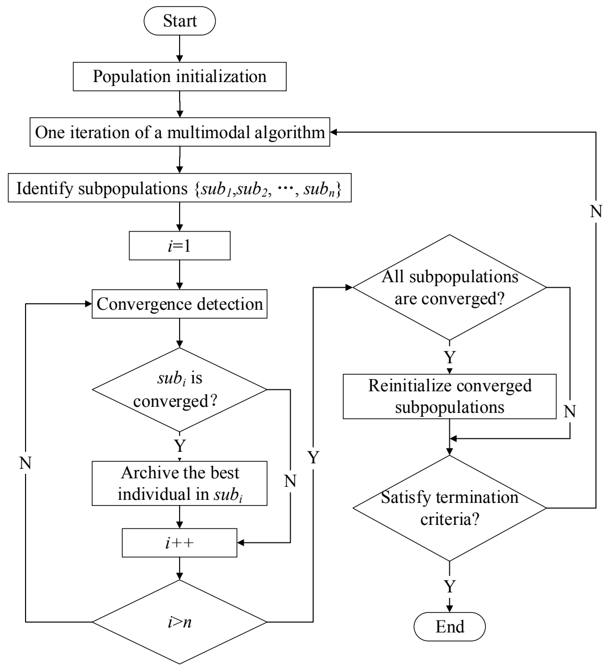

Start
Population initialization
One iteration of a multimodal algorithm
Identify subpopulations {sub1,sub2, ···, subn}
i=1
Convergence detection
subi is converged?
Y
N
Archive the best individual in subi
i++
i>n
Y
N
All subpopulations are converged?
Y
Reinitialize converged subpopulations
N
Satisfy termination criteria?
Y
N
End

Fig. 2. Flowchart of the archived integrated multimodal algorithm

To track changes in CVF, we record the minimal CVF (denoted by minCVF) of the subpopulation over iterations.

Given a subpopulation, it is considered as converged when both of the following criteria are satisfied:

1) The minCVF of the subpopulation has not changed for *niter* successive iterations.

2) The best fitness value has not changed for *niter* successive iterations.

In the second criterion, the best fitness value of the subpopulation is used as another indicator to show whether individuals are moving towards a promising area. The two criteria are used together to determine whether the subpopulation is converged. Once a subpopulation is identified as converged, the best individual in the subpopulation is then stored in an archive.

C. Integrated with state-of-the-art PSO-based Multimodal Algorithms

The term 'individual' in this section is referred to as particle in PSO. Each particle records its current position and historical best position (pbest). Since a particle's pbest is much more stable than its current position, each particle is represented by its pbset. We integrate the proposed archive technique with several state-of-the-art PSO-based multimodal algorithms, i.e., FERPSO, r2pso, r3pso, and LIPS. The resulting algorithms are denoted by FERPSO_ar, r2pso_ar, r3pso_ar, and LIPS_ar respectively. Fig. 2 shows the flowchart of the archive-integrated algorithms. Note that to save computational resources, individuals in converged subpopulations will not take part in the evolutionary process of the multimodal algorithms.

TABLE I
TEST FUNCTIONS

No.	Function	Name	Dim	No. global optima
1	F_1	Five-Uneven-Peak Trap	1	2
2	F_2	Equal Maxima	1	5
3	F_3	Uneven Decreasing Maxima	1	1
4	F_4	Himmelblau	2	4
5	F_5	Six-hump Camel Back	2	2
6	F_6	Shubert	2	18
7	F_7	Vincent	2	36
8	F_6	Shubert	3	81
9	F_7	Vincent	3	216
10	F_8	Modified Rastrigin	2	12
11	F_9	Composition Function 1	2	6
12	F_{10}	Composition Function 2	2	8
13	F_{11}	Composition Function 3	2	6
14	F_{11}	Composition Function 3	3	6
15	F_{12}	Composition Function 4	3	8
16	F_{11}	Composition Function 3	5	6
17	F_{12}	Composition Function 4	5	8
18	F_{11}	Composition Function 3	10	6
19	F_{12}	Composition Function 4	10	8
20	F_{12}	Composition Function 4	20	8

TABLE II
MAXFES USED FOR 3 RANGES OF TEST FUNCTIONS

Range of functions	MaxFEs
F_1 to F_5 (1D or 2D)	5.00E+04
F_6 to F_{11} (2D)	2.00E+05
F_6 to F_{12} (3D or higher)	4.00E+05

IV. EXPERIMENTS

In this section, the archive-integrated algorithms are tested on a benchmark function set and compared with the original multimodal algorithms to investigate the effect of the proposed archive technique.

A. Experimental Setup

1) Test Functions: We use the benchmark function set proposed recently in the "IEEE CEC'2013 Special Session and Competition on Niching Methods for Multimodal Optimization" [12]. The benchmark function set contains 12 functions with different characteristics. They are listed in Table I. F_1-F_5 are simple low dimensional multimodal functions. These functions are not scalable. F_6-F_{12} are scalable multimodal functions. For F_6 and F_7, the number of global optima of is determined by its dimension. For F_8-F_{12}, the number of global optima is independent of ithe dimension. F_9-F_{12} are non-symmetric composition functions constructed by several basic functions. Among them, F_9 and F_{10} are separable while F_{11} and F_{12} are non-separable. More detailed descriptions about these functions can be found in [12].

2) Parameter Settings: The parameters of FERPSO, r2pso, r3pso, LIPS and the corresponding archive-integrated multimodal algorithms are set according to the papers [21]-[23]. In the literature, the population size is commonly set according to the number of optima of the problem being solved. We assume here that the numbers of optima of the test functions are unknown and use the default population size setting as in [12]. The population sizes of all algorithms are set to 100. The parameter *k* used in subpopulation identification is set to 6. The other parameter *niter* in convergence detection is set empirically as *niter*=10.

3) Performance Measure: The performance of the multimodal algorithms and their archive versions is compared using two well-known measures, namely the *peak ratio* (PR) and the *success rate* (SR). To compute PR and SR, we first specify a level of accuracy ε, a threshold value under which we consider a global optimum is found. Then, given a fixed maximum number of function evaluations (MaxFEs), we run a multimodal algorithm NR times. The PR and SR of the algorithm are computed as follows:

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NKP \cdot NR} \tag{5}$$

$$SR = \frac{NSR}{NR} \tag{6}$$

where NKP is the number of known global optima. NPF_i denotes the number of global optima found in the i-th run.

TABLE III
EXPERIMENTAL RESULTS OF FERPOS AND FERPSO_AR

Level of	1.00E-01				1.00E-02				1.00E-03				1.00E-04				1.00E-05			
accuracy	FERPSO_ar		FERPSO		FERPSO_ar		FERPSO		FERPSO_ar		FERPSO		FERPSO_ar		FERPSO		FERPSO_ar		FERPSO	
Function	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_1(1D)	**1.000**	**1.000**	0.500	0.000	**1.000**	**1.000**	0.500	0.000	**1.000**	1.000	0.500	0.000	**1.000**	**1.000**	0.500	0.000	**1.000**	**1.000**	0.500	0.000
F_2(1D)	**1.000**	**1.000**	0.992	0.960	**1.000**	**1.000**	0.992	0.960	**1.000**	1.000	0.992	0.960	**1.000**	**1.000**	0.992	0.960	**1.000**	**1.000**	0.976	0.920
F_3(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_4(2D)	1.000	1.000	1.000	1.000	**1.000**	**1.000**	0.930	0.760	**1.000**	**1.000**	0.730	0.200	**1.000**	**1.000**	0.720	0.160	**1.000**	**1.000**	0.720	0.160
F_5(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	**1.000**	**1.000**	0.980	0.960
F_6(2D)	**0.991**	**0.840**	0.453	0.000	**0.991**	**0.840**	0.398	0.000	**0.989**	**0.800**	0.344	0.000	**0.933**	**0.320**	0.313	0.000	0.000	0.000	0.000	0.000
F_7(2D)	**0.788**	**0.080**	0.174	0.000	**0.737**	0.000	0.174	0.000	**0.712**	0.000	0.166	0.000	**0.670**	0.000	0.156	0.000	**0.606**	0.000	0.147	0.000
F_6(3D)	**0.518**	0.000	0.074	0.000	**0.459**	0.000	0.066	0.000	**0.380**	0.000	0.059	0.000	**0.306**	0.000	0.057	0.000	**0.259**	0.000	0.056	0.000
F_7(3D)	**0.346**	0.000	0.032	0.000	**0.319**	0.000	0.032	0.000	**0.295**	0.000	0.028	0.000	**0.259**	0.000	0.024	0.000	**0.222**	0.000	0.021	0.000
F_8(2D)	**1.000**	**1.000**	0.713	0.040	**1.000**	**1.000**	0.683	0.040	**1.000**	**1.000**	0.650	0.040	**1.000**	**1.000**	0.597	0.000	**1.000**	**1.000**	0.537	0.000
F_9(2D)	**0.940**	**0.720**	0.387	0.000	**0.680**	0.000	0.327	0.000	**0.673**	0.000	0.300	0.000	**0.673**	0.000	0.293	0.000	**0.667**	0.000	0.293	0.000
F_{10}(2D)	**0.770**	0.000	0.245	0.000	**0.750**	0.000	0.245	0.000	**0.680**	0.000	0.245	0.000	**0.645**	0.000	0.230	0.000	**0.605**	0.000	0.230	0.000
F_{11}(2D)	**0.707**	**0.040**	0.427	0.000	**0.673**	0.000	0.387	0.000	**0.673**	0.000	0.373	0.000	**0.673**	0.000	0.367	0.000	**0.673**	0.000	0.360	0.000
F_{11}(3D)	**0.933**	**0.720**	0.267	0.000	**0.667**	0.000	0.267	0.000	**0.667**	0.000	0.267	0.000	**0.647**	0.000	0.260	0.000	**0.640**	0.000	0.253	0.000
F_{12}(3D)	**0.490**	**0.080**	0.105	0.000	**0.390**	0.000	0.105	0.000	**0.370**	0.000	0.105	0.000	**0.355**	0.000	0.105	0.000	**0.345**	0.000	0.105	0.000
F_{11}(5D)	**0.627**	**0.360**	0.140	0.000	**0.407**	0.000	0.140	0.000	**0.393**	0.000	0.140	0.000	**0.387**	0.000	0.140	0.000	**0.373**	0.000	0.140	0.000
F_{12}(5D)	**0.215**	**0.080**	0.025	0.000	**0.140**	0.000	0.025	0.000	**0.140**	0.000	0.025	0.000	**0.140**	0.000	0.025	0.000	**0.140**	0.000	0.025	0.000
F_{11}(10D)	**0.087**	0.000	0.000	0.000	**0.087**	0.000	0.000	0.000	**0.087**	0.000	0.000	0.000	**0.087**	0.000	0.000	0.000	**0.087**	0.000	0.000	0.000
F_{12}(10D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(20D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE IV
EXPERIMENTAL RESULTS OF R2PSO AND R2PSO_AR

Level of	1.00E-01				1.00E-02				1.00E-03				1.00E-04				1.00E-05			
accuracy	r2pso_ar		r2pso		r2pso_ar		r2pso		r2pso_ar		r2pso		r2pso_ar		r2pso		r2pso_ar		r2pso	
Function	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_1(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_2(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_3(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_4(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_5(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_6(2D)	**0.996**	**0.920**	0.718	0.000	**0.991**	**0.880**	0.687	0.000	**0.987**	**0.800**	0.647	0.000	**0.956**	**0.480**	0.604	0.000	0.000	0.000	0.000	0.000
F_7(2D)	**1.000**	**1.000**	0.489	0.000	**0.813**	0.000	0.457	0.000	**0.783**	0.000	0.430	0.000	**0.754**	0.000	0.417	0.000	**0.731**	0.000	0.392	0.000
F_6(3D)	**0.066**	0.000	0.039	0.000	**0.048**	0.000	0.027	0.000	**0.032**	0.000	0.018	0.000	**0.024**	0.000	0.012	0.000	**0.019**	0.000	0.008	0.000
F_7(3D)	**0.811**	0.000	0.149	0.000	**0.413**	0.000	0.113	0.000	**0.346**	0.000	0.088	0.000	**0.286**	0.000	0.067	0.000	**0.231**	0.000	0.049	0.000
F_8(2D)	**1.000**	**1.000**	0.973	0.720	**1.000**	**1.000**	0.970	0.680	**1.000**	**1.000**	0.967	0.640	**1.000**	**1.000**	0.967	0.640	**1.000**	**1.000**	0.960	0.600
F_9(2D)	**1.000**	**1.000**	0.753	0.080	**0.820**	**0.200**	0.713	0.000	**0.733**	0.000	0.693	0.000	**0.720**	0.000	0.687	0.000	**0.707**	0.000	0.687	0.000
F_{10}(2D)	**0.965**	**0.720**	0.670	0.000	**0.760**	**0.040**	0.640	0.000	**0.705**	0.000	0.610	0.000	**0.675**	0.000	0.595	0.000	**0.635**	0.000	0.580	0.000
F_{11}(2D)	**0.940**	**0.680**	0.680	0.000	**0.673**	0.000	0.667	0.000	**0.667**	0.000	0.660	0.000	**0.667**	0.000	0.660	0.000	**0.667**	0.000	0.660	0.000
F_{11}(3D)	**1.000**	**1.000**	0.420	0.000	**0.667**	0.000	0.340	0.000	**0.667**	0.000	0.273	0.000	**0.640**	0.000	0.193	0.000	**0.540**	0.000	0.153	0.000
F_{12}(3D)	**0.780**	**0.120**	0.180	0.000	**0.515**	0.000	0.125	0.000	**0.450**	0.000	0.095	0.000	**0.350**	0.000	0.080	0.000	**0.265**	0.000	0.055	0.000
F_{11}(5D)	**0.020**	0.000	0.000	0.000	**0.013**	0.000	0.000	0.000	**0.007**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(5D)	0.005	0.000	**0.010**	0.000	0.005	0.000	**0.010**	0.000	0.000	0.000	**0.005**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{11}(10D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(10D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(20D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE V
EXPERIMENTAL RESULTS OF R3PSO AND R3PSO_AR

Level of	1.00E-01				1.00E-02				1.00E-03				1.00E-04				1.00E-05			
accuracy	r3pso_ar		r3pso		r3pso_ar		r3pso		r3pso_ar		r3pso		r3pso_ar		r3pso		r3pso_ar		r3pso	
Function	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_{1}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{2}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{3}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{4}(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{5}(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{6}(2D)	**1.000**	**1.000**	0.744	0.000	**1.000**	**1.000**	0.744	0.000	**0.998**	**0.960**	0.744	0.000	**0.996**	**0.920**	0.738	0.000	0.000	0.000	0.000	0.000
F_{7}(2D)	**0.998**	**0.960**	0.381	0.000	**0.780**	0.000	0.381	0.000	**0.758**	0.000	0.378	0.000	**0.716**	0.000	0.378	0.000	**0.694**	0.000	0.374	0.000
F_{6}(3D)	**0.640**	0.000	0.124	0.000	**0.605**	0.000	0.116	0.000	**0.576**	0.000	0.109	0.000	**0.555**	0.000	0.100	0.000	**0.532**	0.000	0.095	0.000
F_{7}(3D)	**0.645**	0.000	0.104	0.000	**0.438**	0.000	0.102	0.000	**0.408**	0.000	0.097	0.000	**0.383**	0.000	0.093	0.000	**0.363**	0.000	0.089	0.000
F_{8}(2D)	**1.000**	**1.000**	0.950	0.480	**1.000**	**1.000**	0.950	0.480	**1.000**	**1.000**	0.950	0.480	**1.000**	**1.000**	0.950	0.480	**1.000**	**1.000**	0.950	0.480
F_{9}(2D)	**1.000**	**1.000**	0.720	0.000	**0.920**	**0.560**	0.700	0.000	**0.913**	**0.520**	0.700	0.000	**0.913**	**0.520**	0.700	0.000	**0.900**	**0.440**	0.700	0.000
F_{10}(2D)	**0.985**	**0.880**	0.690	0.000	**0.915**	**0.400**	0.680	0.000	**0.850**	**0.200**	0.680	0.000	**0.835**	**0.160**	0.670	0.000	**0.800**	**0.160**	0.665	0.000
F_{11}(2D)	**0.813**	**0.280**	0.647	0.000	**0.707**	0.000	0.647	0.000	**0.693**	0.000	0.647	0.000	**0.687**	0.000	0.647	0.000	**0.687**	0.000	0.647	0.000
F_{11}(3D)	**1.000**	**1.000**	0.593	0.000	**0.673**	0.000	0.560	0.000	**0.667**	0.000	0.560	0.000	**0.667**	0.000	0.553	0.000	**0.667**	0.000	0.547	0.000
F_{12}(3D)	**0.875**	**0.400**	0.270	0.000	**0.620**	0.000	0.255	0.000	**0.605**	0.000	0.255	0.000	**0.575**	0.000	0.250	0.000	**0.550**	0.000	0.235	0.000
F_{11}(5D)	**0.900**	**0.720**	0.087	0.000	**0.373**	0.000	0.060	0.000	**0.220**	0.000	0.040	0.000	**0.147**	0.000	0.033	0.000	**0.080**	0.000	0.027	0.000
F_{12}(5D)	**0.345**	0.000	0.070	0.000	**0.175**	0.000	0.055	0.000	**0.125**	0.000	0.050	0.000	**0.065**	0.000	0.045	0.000	**0.045**	0.000	0.015	0.000
F_{11}(10D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(10D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F_{12}(20D)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE VI
EXPERIMENTAL RESULTS OF LIPS AND LIPS_AR

Level of	1.00E-01				1.00E-02				1.00E-03				1.00E-04				1.00E-05			
accuracy	LIPS_ar		LIPS		LIPS_ar		LIPS		LIPS_ar		LIPS		LIPS_ar		LIPS		LIPS_ar		LIPS	
Function	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F_{1}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{2}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{3}(1D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{4}(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{5}(2D)	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
F_{6}(2D)	**0.991**	**0.840**	0.762	0.000	**0.980**	**0.680**	0.760	0.000	**0.969**	**0.520**	0.753	0.000	**0.947**	0.320	0.727	0.000	0.000	0.000	0.000	0.000
F_{7}(2D)	**1.000**	**1.000**	0.494	0.000	**0.788**	0.000	0.494	0.000	**0.687**	0.000	0.493	0.000	**0.608**	0.000	0.491	0.000	**0.543**	0.000	0.488	0.000
F_{6}(3D)	**0.608**	0.000	0.217	0.000	**0.582**	0.000	0.214	0.000	**0.568**	0.000	0.207	0.000	**0.558**	0.000	0.200	0.000	**0.549**	0.000	0.194	0.000
F_{7}(3D)	**1.000**	**0.960**	0.126	0.000	**0.372**	0.000	0.126	0.000	**0.266**	0.000	0.126	0.000	**0.225**	0.000	0.125	0.000	**0.202**	0.000	0.124	0.000
F_{8}(2D)	**1.000**	**1.000**	0.997	0.960	**1.000**	**1.000**	0.997	0.960	**1.000**	**1.000**	0.997	0.960	**1.000**	**1.000**	0.997	0.960	**1.000**	**1.000**	0.997	0.960
F_{9}(2D)	**1.000**	**1.000**	0.867	0.240	0.807	**0.200**	**0.840**	0.160	0.753	0.120	**0.840**	**0.160**	0.727	0.040	**0.827**	**0.160**	0.713	0.040	**0.827**	**0.160**
F_{10}(2D)	**0.890**	0.200	0.845	0.200	0.830	0.000	0.830	**0.160**	0.760	0.000	**0.820**	**0.160**	0.740	0.000	**0.810**	**0.160**	0.730	0.000	**0.810**	**0.160**
F_{11}(2D)	**0.893**	**0.520**	0.693	0.000	0.680	0.000	**0.693**	0.000	0.667	0.000	**0.693**	0.000	0.667	0.000	**0.687**	0.000	0.667	0.000	**0.687**	0.000
F_{11}(3D)	**1.000**	**1.000**	0.627	0.000	**0.673**	0.000	0.627	0.000	**0.667**	0.000	0.627	0.000	**0.667**	0.000	0.620	0.000	**0.667**	0.000	0.620	0.000
F_{12}(3D)	**0.915**	**0.600**	0.410	0.000	**0.635**	0.000	0.410	0.000	**0.635**	0.000	0.405	0.000	**0.630**	0.000	0.405	0.000	**0.630**	0.000	0.395	0.000
F_{11}(5D)	**0.947**	**0.840**	0.187	0.000	**0.627**	0.000	0.180	0.000	**0.620**	0.000	0.180	0.000	**0.600**	0.000	0.180	0.000	**0.593**	0.000	0.180	0.000
F_{12}(5D)	**0.720**	**0.240**	0.200	0.000	**0.530**	0.000	0.200	0.000	**0.520**	0.000	0.200	0.000	**0.505**	0.000	0.200	0.000	**0.500**	0.000	0.200	0.000
F_{11}(10D)	**0.507**	**0.120**	0.040	0.000	**0.340**	0.000	0.040	0.000	**0.340**	0.000	0.040	0.000	**0.333**	0.000	0.040	0.000	**0.333**	0.000	0.040	0.000
F_{12}(10D)	**0.195**	0.000	0.005	0.000	**0.185**	0.000	0.005	0.000	**0.175**	0.000	0.005	0.000	**0.170**	0.000	0.005	0.000	**0.170**	0.000	0.005	0.000
F_{12}(20D)	**0.150**	0.000	0.000	0.000	**0.085**	0.000	0.000	0.000	**0.085**	0.000	0.000	0.000	**0.085**	0.000	0.000	0.000	**0.080**	0.000	0.000	0.000

NSR is the number of successful runs. A successful run is a run in which all global optima are found. The settings of MaxFEs for the test functions are shown in Table II. Five levels of accuracy {1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5} are used to evaluate the performance of a multimodal algorithm under different accuracy requirements. Each multimodal algorithm is run 30 times for each test function.

B. *Overall Performance*

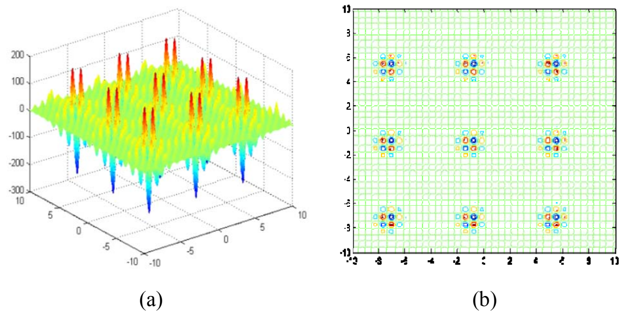
Tables III-VI present the experimental results of the multimodal algorithms on the CEC2013 benchmark function set. In each table, a multimodal algorithm is compared with its corresponding archive-integrated version. The better results are highlighted in boldface. From the tables, it can be seen that the archive-integrated algorithms perform better than the original algorithms in majority of the test functions in terms of peak ratio, especially when the required accuracy level is low (1.0E-1 and 1.0E-2). Moreover, the utilization of archive also increases the success rate of the multimodal algorithms on some test functions. For low dimensional problems with a few optima (F_1-F_5), r2pso, r3pso, and LIPS are capable of achieving 100% peak ratio and success rate regardless of the accuracy level. However, when dealing with F_6(2D, 3D) and F_7(2D, 3D), which have many global optima, their performance degrades. By using an external archive, the performance of the multimodal algorithms can be improved, as revealed by the results of FERPSO_ar, r2pso_ar, r3pso_ar, and LIPS_ar. It can be noted that the results of r2pso_ar, r3pso_ar, and LIPS_ar on F_6 and F_7 remain very competitive. It can also be noted that all the archive-integrated algorithms perform very well on F8, whose number of optima is 12. They achieve the maximum PR and SR in all levels of accuracy. As for the complex composition functions (F_9-F_{12}), it is hard for all the algorithms to get a high SR. Nevertheless, the use of the archive still improves the multimodal algorithms' performance measured by PR.

C. *Search Behavior*

To further investigate the effect of the proposed archive technique, we carry out another experiment. We examine the search behavior of LIPS and LIPS_ar on F_6(2D). Fig. 3 shows the function plot and contour plot of F_6(2D). We divide the search space into 20×20 girds and count the number of function evaluations spent on each grid through a typical run of each algorithm. The results are shown in Fig. 4. The brightness of a grid corresponds to the proportion of the total number of function evaluations spent on it. The brighter a grid is, the more function evaluations are spent on it. It can be observed that LIPS_ar can search the whole space more evenly than LIPS. This is because by using the proposed archive technique, individuals in converged subpopulations are identified and reinitialized to search for other optima in the space.

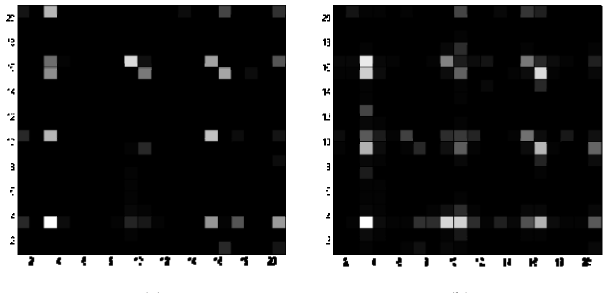
D. *Effect of Population Size*

We finally study the effect of population size on the algorithms' ability to locate multiple global optima. Specifically, we investigate the performance of LIPS and LIPS_ar on two functions that have many global optima (F_6(3D) and F_7(3D)) using different population sizes, i.e., NP={50, 60, …, 200}. The accuracy level is set to 1.0E-3. For each algorithm, we record the average number of optima found



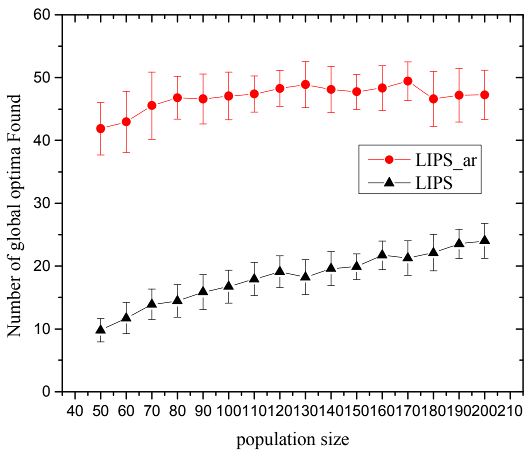
(a) (b)

Fig. 3. Function plot and contour plot of F_6(2D)

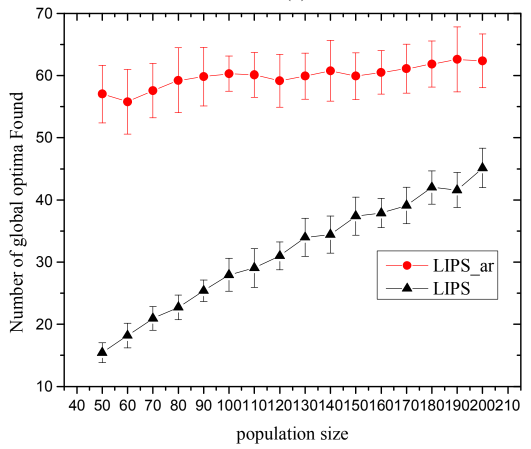


(a) (b)

Fig. 4. Function evaluations spent on each grid (a) LIPS (b)LIPS_ar



(a)



(b)

Fig. 5. Average number of optima found using different population sizes. (a)F_6(3D) (b)F_7(3D)

over 30 independent runs. The experimental results are shown in Fig. 5. It can be seen that LIPS_ar can locate more optima than LIPS. This is because the ability of LIPS to detect a large number of optima is enhanced by reusing the individuals in converged subpopulations. Meanwhile, the use of the archive guarantees that the found optima are maintained throughout the evolutionary process. Most importantly, the performance of LIPS_ar is less sensitive to the population size used.

V. Conclusion

This paper proposed a generic archive technique for multimodal algorithms to reduce the influence of the population size parameter. The proposed archive technique contains two components, subpopulation identification and convergence detection. Converged subpopulations are first identified using the two components. Individuals in the converged subpopulations are then stored in an external archive and are reused to search for other optima. We integrate the archive technique with several state-of-the-art PSO-based multimodal algorithms (FERPSO, r2pso, r3pso, and LIPS). Experimental results on the CEC2013 multimodal function set show that the proposed archive technique can improve the multimodal algorithms' performance and reduce the influence of the population size parameter. For future research, it would be interesting to apply the proposed archive technique to DE-based multimodal algorithms.

REFERENCES

[1] S. Das, S. Maity, B.-Y. Qu, and P. N. Suganthan, “Real-parameter evolutionary multimodal optimization: A survey of the state-of-the-art,” *Swarm Evol. Comput.*, vol. 1, no. 2, pp. 71–88, Jun. 2011.

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

[3] 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.

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

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

[6] 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.

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

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

[9] 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

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

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

[12] X. Li, A. Engelbrecht, and M. G. Epitropakis, “Benchmark functions for cec’2013 special session and competition on niching methods for multimodal function optimization,” Evolutionary Computation and Machine Learning Group, RMIT University, Melbourne, Australia, Tech. Rep., 2013

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

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

[15] 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.

[16] 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.

[17] 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.

[18] I.L. Schoeman and A.P. Engelbrecht, “A Parallel Vector-Based Particle Swarm Optimizer,” in *Proc. of the Int. Conf. on Neural Networks and Genetic Algorithms*, 2005, pp. 268-271.

[19] E. Özcan and M. YÕlmaz, Particle Swarms for Multimodal Optimization, in Adaptive and Natural Computing Algorithms, B. Beliczynski, et al., Editors. 2007, Springer Berlin / Heidelberg. p. 366-375.

[20] J. Barrera and C. Coello, “A Review of Particle Swarm Optimization Methods Used for Multimodal Optimization,” in Innovations in Swarm Intelligence, Springer Berlin Heidelberg, 2009, pp. 9-37.

[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, “Niching without niching parameters: Particle swarm optimization using a ring topology,” *IEEE Trans. Evol. Computat.*, vol. 14, no. 1, pp. 150–169, Feb. 2010.

[23] B. Y. Qu, P. N. Suganthan, and S. Das, “A distance-based locally informed particle swarm model for multi-modal optimization,” *IEEE Trans. Evol. Comput.*, vol. 17, no. 3, pp. 387–402, Jun. 2013.

[24] L. Kauffman and P. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*. J. Wiley & Sons, 1990.