Composite Differential Evolution with Queueing Selection for Multimodal Optimization

Yu-Hui Zhang[1,3,4], Yue-Jiao Gong[2,3,4,*], Wei-Neng Chen[2,3,4] and Jun Zhang[2,3,4]
[1] Department of Computer Science, Sun Yat-sen University
[2] School of Advanced Computing, Sun Yat-sen University, Guangzhou, China
[3] Key Laboratory of Machine Intelligence and Advanced Computing, Ministry of Education
[4] Engineering Research Center of Supercomputing Engineering Software, Ministry of Education
*Corresponding Author: gongyuejiao@gmail.com

***Abstract*—The aim of multimodal optimization is to locate multiple optima of a given problem. Evolutionary algorithms (EAs) are one of the most promising candidates for multimodal optimization. However, due to the use of greedy selection operators, the population of an EA will generally converge to one region of attraction. By incorporating a well-designed selection operator that can facilitate the formation of different species, EAs will be able to allow multiple convergence. Following this research avenue, we propose a novel selection operator, namely, queueing selection (QS) and integrate it with one of the most promising DE variants, called composite differential evolution (CoDE). The integrated algorithm (denoted by CoDE-QS) inherits the strong global search ability of CoDE and is capable of finding and maintaining multiple optima. It has been tested on the CEC2013 benchmark functions. Experimental results show that CoDE-QS is very competitive.**

Keywords—differential evolution; multimodal optimization; niching; clearing;

I. INTRODUCTION

Many optimization problems contain multiple satisfactory solutions. When solving such kind of problems, we are often interested in finding multiple good solutions simultaneously instead of a single solution. The motivation is that multiple good solutions can provide better understanding of the problem landscape. On the other hand, when a solution cannot be realized due to the physical constraints, we can quickly switch to other solutions without causing significant performance loss.

Evolutionary algorithms (EAs) are population-based metaheuristic search algorithms. They have been shown to be very effective in solving various kinds of optimization problems [1][2]. However, EAs are originally designed to locate a single global optimum of a given problem regardless of the problem landscape. The population of an EA will finally converge to one region of attraction. The solutions (individuals) in the final population will be identical or very similar. This is adverse to the intention of finding multiple optima. To make EAs suitable for multimodal optimization, a number of techniques commonly known as "niching" have been proposed [3]. Niching techniques are designed to prevent the population of an EA converges to a single optimum. By applying the niching techniques, multiple species are formed around different basins of attractions. If the species are maintained throughout the running process, the algorithm will be able to provide multiple distinct solutions. Some famous niching techniques include: fitness sharing [4], crowding [5], speciation [6], and clearing [7].

This work was supported in part by the 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 in part by NSFC Joint Fund with Guangdong under Key Projects U1201258.

A. Fitness sharing

Fitness sharing is one of the earliest niching techniques. It is proposed by Holland [8] and later extended by Goldberg and Richardson [4]. The concept is that there are only limited resources in each region of the search space. Individuals within a same region have to share the limited resources with one another. In an EA, sharing was implemented by scaling the fitness of an individual based on the number of 'similar' individuals in the population. A threshold value called sharing radius σ_{share} is used to determine whether two individuals are similar. The shared fitness of the ith individual is calculated as follows:

$$f_{\text{shared}}(i) = \frac{f(i)}{\sum_{j=1}^{NP} sh(d_{ij})} \tag{1}$$

where the sharing function is defined as:

$$sh(d_{ij}) = \begin{cases} 1-(d_{ij}/\sigma_{\text{share}})^{\alpha}, & \text{if } d_{ij} < \sigma_{\text{share}} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

d_{ij} is the distance between individuals i and j. NP is the population size and α is a constant called sharing level. The shared fitness of individual i is greatly reduced if there are highly similar individuals in the population.

B. Crowding

Crowding is a simple niching technique introduced by De Jong [9]. The method tries to maintain the population diversity by limiting the competition to similar individuals. The similarity of two individuals is generally measured by their Euclidean distance. This approach compares an offspring with

a number of sampled individuals from the current population. The most similar individual will be replaced if the offspring has a better fitness value. The number of sampled individuals is controlled by a parameter called crowding factor *CF*. The drawback of crowding is the occurrence of replacement error that dissimilar individuals are sampled and replaced while using a relatively small *CF*. A simple way to fix the problem is by setting *CF* to *NP*. The complexity of crowding goes up when this modification is applied.

C. Speciation

Speciation [6] is another commonly used niching technique. It divides the population into a number of species. The update operations (e.g. crossover and mutation) are then performed in each species instead of the whole population. Like fitness sharing, a radius parameter r_s needs to be specified before the division of individuals. The individuals are first sorted according to their fitness values. The best individual is marked as the species seed of a newly generated species. All other individuals fall within the niche radius of the species seed is identified as the same species. Then, individuals belong to the species are removed and the above steps are repeated to generate new species. The repeated process is terminated when all individuals have been removed. In this way, the whole population is divided into groups according to their similarity.

D. Clearing

The original clearing [7] is applied after the fitness evaluation of individuals and before the selection operator. Like sharing and speciation, it also depends on a radius parameter called clearing radius. Clearing first divides the population into a number of species by using a procedure similar to the speciation. Then the fitness of the dominate individual is preserved while the fitness of other individuals in the same species is set to 0. Clearing maintains the population diversity by inhibiting the growth of similar individuals and giving more resources for individuals that are distinct.

From the above description, it can be observed that the techniques have one thing in common. They modify the original greedy selection operator to a more restricted one to support the formation of multiple species. This indicates that the selection operator plays a very important role in EA-based multimodal optimization algorithms. Over the past decades, a number of promising EA variants have been proposed [10]-[13] to tackle complex problems. This paper tries to exploit the potential of using these algorithms to handle multimodal problems by incorporating a less greedy selection operator. To this end, we propose a queueing selection (QS) operator. The operator is designed to facilitate the maintenance of multiple good solutions, and to preserve the search ability of EAs. It divides the individuals into a number of species. The offspring population is generated by picking individuals from different species. QS is integrated with a recently proposed differential evolution (DE) variant (called CoDE [12]) to make a competitive multimodal algorithm. Effectiveness of the integrated algorithm (CoDE-QS) has been demonstrated by comparing its performance with a number of state-of-the-art multimodal algorithms over the CEC2013 multimodal test suite.

The rest of the paper is organized as follows. Section II gives a brief review of the literature on population-based multimodal algorithms and introduces the CoDE algorithm. The proposed queueing selection operator and its integration with CoDE are detailed in section III. Experiments on the CEC2013 test suite are carried out in Section IV, with thorough analysis of the experimental results. Concluding remarks and future research directions are given in Section V.

II. Related Work

A. Differential Evolution for Multimodal Optimization

Differential evolution (DE), proposed by Storn and Price [14] in 1995, is one of the most popular EAs. It is a simple yet powerful global optimization technique. In the literature, it has been adapted to solve multimodal optimization problems [5],[15]-[17]. Thomsen [5] applied the concepts of fitness sharing and crowding to DE, respectively, and obtained the sharing DE (ShDE) and crowding DE (CDE) algorithms. To eliminate replacement error, *CF* is set to the population size in CDE. When an offspring is generated, it is compared with the nearest individual in the current population, the individual will be replaced if the offspring has a better fitness value.

Species-based DE (SDE), proposed by Li [15], is another famous DE-based multimodal algorithm. It adopts the concept of speciation. Each species is formed around a species seed. If the number of individuals in a species is less than *m*, then local random individuals are generated around the species seed. Crossover and mutation are thereafter performed in each species.

B. Particle Swarm Optimization for Multimodal Algorithms

Particle swarm optimization (PSO) [18][19] is a population-based optimization technique inspired by the social behavior of animals. This subsection introduces three state-of-the-art PSO-based multimodal algorithms compared in this paper. Note that most of existing multimodal 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 [20] introduced a quantity called fitness Euclidean-distance Ratio (FER) and proposed FERPSO. In FERPSO, each particle moves towards its pbest and best neighbor. The best neighbor of a particle is chosen according to FER. By using the neighborhood best instead of gbest, multiple niches are naturally formed around multiple optima. In [21], PSO using a ring topology (rpso) is recommended to tackle multimodal problems. The rpso does not require any niching parameters and is very simple. Particles are arranged in a circle and each particle only interacts with its direct neighbors. The ring topology PSO is found to be able to form multiple stable niches. Experimental results reported in [21] show that rpso is very promising in solving multimodal problems. To avoid all particles converge to a single optimum, Qu *et al*. [22] 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.

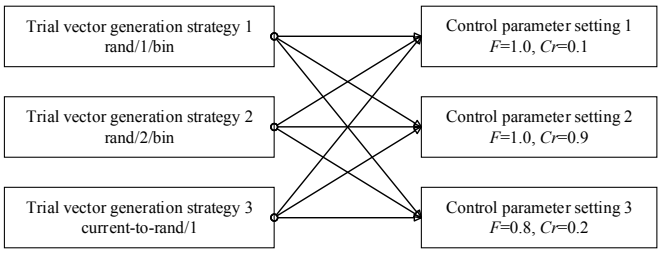


Fig. 1. Illustration of combining trial vector generation strategies with control parameter settings

C. Composite Differential Evolution

DE uses mutation, crossover, and selection at each generation to move its population toward the global optimum. The performance of DE depends on two of its components, i.e., trial vector generation strategy (mutation and crossover) and setting of control parameters (scaling factor F, crossover rate Cr, and population size NP). In most DE variants, only one trial vector generation strategy and one control parameter setting are employed at each generation. However, recent research indicates that the ensemble of multiple strategies is very promising [23]-[24]. Motivated by the finding that DE's performance can be improved by combining several effective trial vector generation strategies with some suitable control parameter settings, Wang *et al.* [12] proposed a novel method called composite DE (CoDE). CoDE randomly combines three trial vector generation strategies and three control parameter settings. The idea of CoDE is illustrated in Fig. 1.

The trial vector generation strategies and control parameter settings of CoDE are chosen in a way that they have distinct advantages. Therefore, their combination can be effective in solving different kind of problems. The three trial vector generation strategies are:

1) rand/1/bin
2) rand/2/bin
3) current-to-rand/1

The strategies are shown in equations at the bottom of this page. In the equations, $x_{i,j}$ represents the jth dimension of the ith individual. $r1$, $r2$, $r3$, $r4$, and $r5$ are distinct integers randomly chosen from the range [1, NP] and are different from i. j_{rand} is a random integer in [1, D], where D is the dimension of the problem being solved. The three control parameter settings are:

1) [F=1.0, Cr=0.1]
2) [F=1.0, Cr=0.9]
3) [F=0.8, Cr=0.2]

The features of the trial vector generation strategies are as follows. "rand/1/bin" is a commonly used strategy in the literature. In "rand/1/bin", all vectors for mutation are selected from the population at random. Therefore, there is no bias towards any search direction. In "rand/2/bin", two differential vectors are added to the based vector instead of one. This strategy can generate more different trial vectors than "rand/1/bin". "current-to-rand/1" uses the arithmetic crossover rather than the binomial crossover [25]. It is a rotation-invariant strategy and suitable for rotated problems. Similarly, the control parameter settings have distinct features. [F=1.0, Cr=0.1] is suitable for dealing with separable problems. [F=1.0, Cr=0.9] is used to maintain the population diversity and to make the three strategies more powerful in global exploration. [F=0.8, Cr=0.2] encourages the exploitation of the three strategies in the search space and accelerates the convergence speed of the population.

For each individual, the three strategies are used to create three trial vectors with control parameter settings randomly chosen from the parameter candidate pool. The best of the three trial vectors will enter the next generation if it is better than the individual. The experimental results reported in [12] show that CoDE is superior to a number of state-of-the-art DE algorithms when solving complex problems.

III. CoDE with Queueing Selection

The selection operator of CoDE is based on simple competitions between individuals and their corresponding trial vectors (offspring). It is suitable for finding a single global optimum. However, on the other hand, it limits the capability of CoDE of tracing and maintaining multiple optima. To make CoDE an eligible multimodal algorithm, we introduce a queueing selection (QS) operator and integrate it into CoDE. The resulting algorithm is called CoDE-QS, which is detailed in the rest of this section.

A. Queueing Selection

The clearing procedure proposed by Pétrowski[7] is conducted before the selection operator. Individuals are divided into species according to their similarity. For each species, the clearing procedure preserves the fitness of the dominant individual (species seed) and resets the fitness of all other individuals to 0. In this way, the computational resource

"rand/1/bin"

$$u_{i,j} = \begin{cases} x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j}), & \text{if } rand < C_r \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (3)$$

"rand/2/bin"

$$u_{i,j} = \begin{cases} x_{r1,j} + F \cdot (x_{r2,j} - x_{r3,j}) + F \cdot (x_{r4,j} - x_{r5,j}), & \text{if } rand < C_r \text{ or } j = j_{rand} \\ x_{i,j} & \text{otherwise} \end{cases} \quad (4)$$

"current-to-rand/1"

$$u_{i,j} = x_{i,j} + rand \cdot (x_{r1,j} - x_{i,j}) + F \cdot (x_{r2,j} - x_{r3,j}) \quad (5)$$

is allocated to the dominant individual. This favors the maintenance of population diversity. However, note that the search ability of an EA owes much to the cooperation of individuals, the mechanism also weakens the local search ability. It is possible to generalize the clearing procedure by accepting K survivors in each species. The drawback is that it introduces an additional parameter K. The proposed queueing selection operator is an adaptation of the clearing procedure that tries to eliminate the above problems. During the selection process, we maintain a species list. Each species is implemented as a queue of individuals. The procedure of queueing selection is as follows:

Step 1. Create an empty list of species *LS* and a list of individuals *LI*. Append the individuals in the current population and their offspring (trial vectors) to *LI*. Sort *LI* in decreasing order according to the individuals' fitness values.

Step 2. Create a new species Q_i. Pick the first individual X_0 from *LI* and append it to Q_i. Go through the rest of *LI*. For each individual X in *LI*, calculate the distance d between X_0 and X. If d is less than a threshold value σ (niche radius), X is removed from *LI* and appended to Q_i.

Step 3. Append Q_i to *LS*.

Step 4. Repeat Step 2 and Step 3 until *LI* is empty.

Step 5. Go through the list of species *LS*. For a nonempty species Q_i, remove the individual X_h at the head and add X_h to the population of the next generation.

Step 6. Repeat Step 5 until there are *NP* individuals selected.

Fig. 2 illustrates the queueing selection process. Each rectangle represents a species. The circles filled with patterns are used to denote the individuals. The chain of arrow lines gives the directions of picking individuals. It successively picks individuals from the species instead of only extracting the species seeds. There are two advantages of queueing selection to solve multimodal problems. (1)Allow several individuals to survive in each species without introducing additional parameters. (2)Strike a balance between the ability for diversity maintenance and the ability for local search and fine-tuning. The population diversity is preserved by picking individuals from different species. Moreover, the quality of selected individuals is guaranteed by choosing top-ranking individuals in each species (note that individuals in each species have been arranged in order of decreasing fitness because of the sorting of *LI* in Step 1).

B. CoDE with Queueing Selection

The queueing selection is integrated with CoDE to make a competitive multimodal algorithm. The resulting algorithm is called CoDE-QS. For each individual in the population, CoDE-QS generates three trial vectors using the three complementary strategies. Queueing selection is conducted after all the trial vectors have been generated. First, individuals in the current population and their corresponding trial vectors are added to *LI*. Then CoDE-QS goes through the queueing selection steps to obtain the population for the next generation. The pseudo of CoDE-QS is given in **Algorithm 1**.

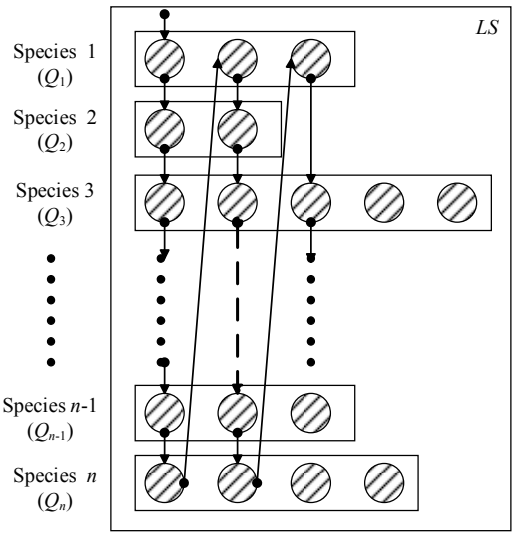


Fig. 2. Illustration of queueing selection.

Algorithm 1 CoDE-QS

1: G=0;
2: Generate an initial population P_0={X_1, X_2, …, X_{NP}} by randomly sampling from the search space;
3: evaluate the fitness values of the individuals;
4: FEs=NP;
5: **while** FEs<$MaxFEs$ **do**
6: U_G=∅;
7: **for** X_i in P_G **do**
8: Use the three trial vector generation strategies, each with a control parameter setting randomly selected from the parameter candidate pool, to generate three trial vectors: $U_{i,1}$, $U_{i,2}$, $U_{i,3}$
9: evaluate the fitness values of the trial vectors;
10: FEs=FEs+3;
11: U_G= U_G∪{$U_{i,1}$, $U_{i,2}$, $U_{i,3}$};
12: **end for**
13: Use queueing selection to obtain P_{G+1} from P_G ∪U_G;
14: G=G+1;
15: **end while**

The complexity of generating trial vectors is $O(D{\cdot}NP)$, where NP is the population size, D is the encoding length of individuals. The complexity of the selection process is composed of several parts. In step 1, sort the individuals in *LI* takes $O(NP\log NP)$ time. The complexity of steps 2-4 can be estimated by the number of distance calculations, which is approximately $O(D{\cdot}NP^2)$. Lastly, steps 5-6 take $O(NP)$ time. Therefore, the overall complexity of CoDE-QS is $O(D{\cdot}NP^2)$, which is similar to that of SDE, CDE and ShDE.

IV. EXPERIMENTS

In this section, we carry out experiments to investigate the performance of the proposed algorithm. Specially, CoDE-QS is tested on a recently proposed benchmark function set and compared with some state-of-the-art multimodal algorithms.

TABLE I
TEST FUNCTIONS

Function	Name	Dim	#global optima	r
F_1	Five-Uneven-Peak Trap	1	2	0.01
F_2	Equal Maxima	1	5	0.01
F_3	Uneven Decreasing Maxima	1	1	0.01
F_4	Himmelblau	2	4	0.01
F_5	Six-hump Camel Back	2	2	0.5
F_6	Shubert	2	18	0.5
F_7	Vincent	2	36	0.2
F_6	Shubert	3	81	0.5
F_7	Vincent	3	216	0.2
F_8	Modified Rastrigin	2	12	0.01
F_9	Composition Function 1	2	6	0.01
F_{10}	Composition Function 2	2	8	0.01
F_{11}	Composition Function 3	2	6	0.01
F_{11}	Composition Function 3	3	6	0.01
F_{12}	Composition Function 4	3	8	0.01
F_{11}	Composition Function 3	5	6	0.01
F_{12}	Composition Function 4	5	8	0.01
F_{11}	Composition Function 3	10	6	0.01
F_{12}	Composition Function 4	10	8	0.01
F_{12}	Composition Function 4	20	8	0.01

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

A. Experimental Setup

1) Test Functions

The benchmark function set [26] for CEC'2013 competition on niching methods is adopted in the experiment to compare the performance of different multimodal algorithms. The function set contains 12 multimodal functions with different characteristics. They are tabulated in table I. All the functions are formulated as maximization problems. F_1-F_8 are well-known, widely used functions. In this subset, F_1-F_5 are non-scalable 1D or 2D multimodal functions. F_6-F_8 are scalable multimodal functions. The number of optima of F_6 and F_7 is determined by the dimension D. F_9-F_{12} are scalable, non-symmetric multimodal functions constructed by combining several basic functions. F_9 and F_{10} are separable while F_{11} and F_{12} are not. More detailed descriptions of the multimodal functions can be found in [26].

2) Algorithms Conpared and Parameter Settings

CoDE-QS is compared with the following multimodal optimization algorithms:

1) crowding DE (CDE) [5]
2) fitness sharing DE (ShDE) [5]
3) speciation-based DE (SDE) [15]
4) fitness-Euclidean distance ratio PSO (FERPSO) [20]
5) ring topology PSO (r2pso, r3pso) [21]
6) locally informed PSO (LIPS) [22]

All the algorithms are implemented using C++ and executed on a computer with an Intel(R) Core(TM) i3-3240 CPU and 4GB of memory. The parameters of the algorithms are set according to the corresponding papers. The default population size setting used in [26] is adopted. That is, the population sizes of the algorithms are fixed at 100 (NP=100). The settings of niche radius of SDE, ShDE, CoDE-QS are listed in the last column of table I. All algorithms terminate after reaching the maximum number of function evaluations (MaxFEs). The settings of MaxFEs for the test functions are given in table II.

3) Performance Measures

Two commonly used measures, *peak ratio* (PR) and *success rate* (SR), are adopted to evaluate the performance of the multimodal algorithms. PR is the percentage of successfully located optima. SR is the percentage of runs in which all optima are successfully located. To compute PR and SR, an accuracy level ε needs to be specified. If the difference from a computed solution to a known global optimum is below ε, the optimum is considered to have been found. For simple low dimensional function (F_1-F_8), ε is set to 0.0001. For complex composition functions (F_9-F_{12}), ε is set to 0.1. After given the accuracy level, PR and SR are calculated according to (6) and (7) respectively.

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NKP \cdot NR} \tag{6}$$

$$SR = \frac{NSR}{NR} \tag{7}$$

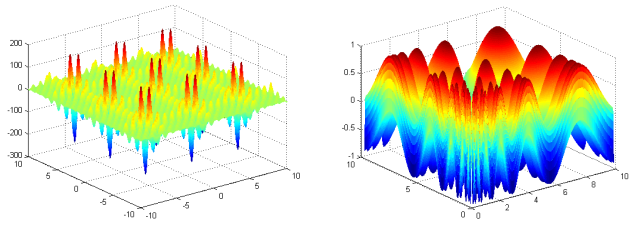
where NPF_i is the number of optima located in ith run. NKP is the total number of optima. In (7), NSR is the number of runs in which all optima are successfully located. NR is the number of runs. In the experiment, each algorithm is run 50 times for each test function (NR=50).

B. Overall Performance

The experimental results of the algorithms are tabulated in Table III. The best results are marked in boldface. From the table, it can be observed that CoDE-QS performs very well on most of the test functions in terms of both PR and SR. The performance of CoDE-QS comes from the search ability of CoDE and the ability of diversity maintenance of queueing selection. To demonstrate the CoDE-QS's capability of locating multiple optima, we visualize the results on F_6(2D) and F_7(2D). F_6 and F_7 are functions with many optima. The global optima of F_6 can be divided into nine groups. Each group contains two optima that are very close to each other. F_7 has peaks that are of different shapes and sizes. The peaks are spread unevenly over the search space. Fig. 3 shows the landscape of F_6(2D) and F_7(2D). They are very challenging multimodal problems. The final population of CoDE-QS is depicted in Fig. 4. It can be seen that CoDE-QS can not only locate all the global optima, but also local optima. For F_7(2D), CoDE-QS also successfully locates all optima.

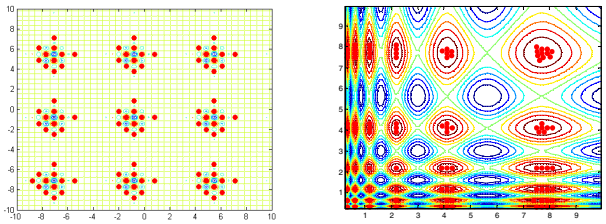
TABLE III
EXPERIMENTAL RESULTS ON CEC2013 TEST SUITE

Alg.	r2pso		r3pso		LIPS		FERPSO		CDE		SDE		ShDE		CoDE-QS	
Function	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**	0.500	0.000	**1.000**	**1.000**	0.580	0.360	0.500	0.000	**1.000**	**1.000**
F_2(1D)	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.996	0.980	**1.000**	**1.000**	0.996	0.980	0.360	0.000	**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**
F_4(2D)	**1.000**	**1.000**	0.995	0.980	**1.000**	**1.000**	0.725	0.220	0.965	0.860	0.265	0.000	0.270	0.000	0.745	0.140
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**	0.520	0.040	0.770	0.580
F_6(2D)	0.604	0.000	0.729	0.000	0.732	0.000	0.326	0.000	0.177	0.000	0.489	0.000	0.000	0.000	**1.000**	**1.000**
F_7(2D)	0.416	0.000	0.360	0.000	0.492	0.000	0.169	0.000	0.710	0.000	0.202	0.000	0.029	0.000	**1.000**	**1.000**
F_6(3D)	0.011	0.000	0.118	0.000	0.203	0.000	0.060	0.000	0.541	0.000	0.072	0.000	0.000	0.000	**1.000**	**1.000**
F_7(3D)	0.070	0.000	0.094	0.000	0.123	0.000	0.024	0.000	0.271	0.000	0.025	0.000	0.004	0.000	**0.463**	**0.000**
F_8(2D)	0.962	0.580	0.952	0.460	0.990	0.880	0.623	0.000	1.000	1.000	0.437	0.000	0.032	0.000	**1.000**	**1.000**
F_9(2D)	0.767	0.080	0.710	0.020	0.860	0.180	0.420	0.000	0.937	0.680	0.170	0.000	0.110	0.000	**1.000**	**1.000**
F_{10}(2D)	0.663	0.000	0.703	0.020	**0.860**	**0.200**	0.250	0.000	0.310	0.000	0.135	0.000	0.085	0.000	0.278	0.000
F_{11}(2D)	0.667	0.000	0.640	0.000	0.680	0.000	0.440	0.000	0.760	0.140	0.170	0.000	0.057	0.000	**0.990**	**0.940**
F_{11}(3D)	0.473	0.000	0.593	0.000	0.653	0.000	0.317	0.000	0.693	0.000	0.163	0.000	0.020	0.000	**1.000**	**1.000**
F_{12}(3D)	0.183	0.000	0.273	0.000	0.415	0.000	0.083	0.000	0.673	0.000	0.090	0.000	0.093	0.000	**1.000**	**1.000**
F_{11}(5D)	0.010	0.000	0.077	0.000	0.200	0.000	0.143	0.000	0.700	0.020	0.063	0.000	0.067	0.000	**1.000**	**1.000**
F_{12}(5D)	0.003	0.000	0.063	0.000	0.195	0.000	0.010	0.000	0.723	0.420	0.005	0.000	0.050	0.000	**0.773**	**0.740**
F_{11}(10D)	0.000	0.000	0.000	0.000	0.037	0.000	0.003	0.000	0.523	0.060	0.000	0.000	0.167	0.000	**1.000**	**1.000**
F_{12}(10D)	0.000	0.000	0.000	0.000	0.003	0.000	0.003	0.000	0.000	0.000	0.000	0.000	**0.125**	**0.000**	**0.125**	**0.000**
F_{12}(20D)	0.000	0.000	0.000	0.000	0.003	0.000	0.000	0.000	0.063	0.060	0.000	0.000	0.125	0.000	**1.000**	**1.000**



(a) (b)

Fig. 3. Landscape of F_6(2D) and F_7(2D). (a) F_6(2D) (b) F_7(2D)



(a) (b)

Fig. 4. Final population of CoDE-QS on F_6(2D) and F_7(2D). (a) F_6(2D) (b) F_7(2D)

C. Convergence Speed

In this part, we compare the convergence speed of the algorithms. The convergence speed of an algorithm is measured by the number of function evaluations (FEs) required to locate all optima. To obtain statistically reliable results, the required FEs is averaged over *NR* runs:

$$MeanFEs = \frac{\sum_{i=1}^{NR} FEs_i}{NR}. \quad (8)$$

In (8), FEs_i denotes the number of FEs consumed in the ith run. If the algorithm cannot locate all optima by MaxFEs, then MaxFEs is counted as the required FEs. The accuracy level ε is fixed at 0.0001 and *NR* is set to 50. Noting that CoDE-QS is the only algorithm that can obtain non-zero SR in most of the composition functions, we concentrate on F_1-F_8. The experimental results are tabulated in table IV. The convergence speed of CoDE-QS is not as good as that of CDE, rpso, and LIPS when dealing with low dimensional simple multimodal functions. The reason is the over consume of FEs for simple functions, since CoDE-QS generates three trial vectors for each individual. However, for complex functions with many optima, CoDE-QS is superior. Plots of the number of found optima versus FEs on F_6 and F_7 are presented in Fig. 5. It can be seen that CoDE-QS can locate more optima with fewer FEs.

D. Effect of Population Size

The setting of population size has an impact on the performance of population-based multimodal algorithms. Generally, a large number of optima requires a large population size. In this part, we investigate the effect of population size parameter. Experiments are conducted on functions with many optima. The accuracy level ε is set to 0.0001. For each algorithm, we record the average number of found optima over 50 independent runs when different population sizes are used. The experimental results are shown in Fig. 6. Compared with other algorithms, CoDE-QS is able to find a large number of optima with a relatively small number of individuals. This is because that the combination of generation strategies has widened the search range of each individual, and therefore lessens the required population size. Further, it is noteworthy that there is a wide range of settings in which CoDE-QS performs equally well. This property decreases the difficulty in choosing the suitable population size.

V. CONCLUSION

In this paper, we proposed a queueing selection (QS) operator to enhance EAs' ability of locating multiple optima. QS is designed to preserve the population diversity, and at the same time facilitate the evolution of individuals. This is

TABLE IV
CONVERGENCE SPEED

Alg.		F_1(1D)	F_2(1D)	F_3(1D)	F_4(2D)	F_5(2D)	F_6(2D)	F_7(2D)	F_6(3D)	F_7(3D)	F_8(2D)
r2pso	Mean	**200.00**	1704.00	1014.00	**6022.00**	2814.00	200000.00	200000.00	400000.00	400000.00	59372.00
	St.D.	**0.00**	428.00	483.33	**810.75**	458.70	0.00	0.00	0.00	0.00	83395.65
r3pso	Mean	**200.00**	1848.00	1066.00	6550.00	2726.00	200000.00	200000.00	400000.00	400000.00	107124.00
	St.D.	**0.00**	583.52	564.84	6241.83	449.80	0.00	0.00	0.00	0.00	94609.80
LIPS	Mean	**200.00**	1654.00	1256.00	8878.00	3704.00	200000.00	200000.00	400000.00	400000.00	**31462.00**
	St.D.	**0.00**	454.41	679.16	978.22	787.14	0.00	0.00	0.00	0.00	**49960.32**
FERPSO	Mean	**200.00**	5002.00	**874.00**	41400.00	5536.00	200000.00	200000.00	400000.00	400000.00	200000.00
	St.D.	**0.00**	8387.34	**446.23**	14729.07	9160.61	0.00	0.00	0.00	0.00	0.00
CDE	Mean	**200.00**	3580.00	2664.00	44768.00	15780.00	200000.00	200000.00	400000.00	400000.00	34878.00
	St.D.	**0.00**	1669.61	2508.69	3740.13	1980.12	0.00	0.00	0.00	0.00	2211.18
SDE	Mean	43369.94	**1344.50**	1203.96	50000.00	**1980.12**	200000.00	200000.00	400000.00	400000.00	200000.00
	St.D.	13786.35	**155.14**	222.95	0.00	**230.10**	0.00	0.00	0.00	0.00	0.00
ShDE	Mean	50000.00	32814.00	1422.00	50000.00	13330.00	200000.00	200000.00	400000.00	400000.00	200000.00
	St.D.	0.00	16572.04	1053.05	0.00	7629.76	0.00	0.00	0.00	0.00	0.00
CoDE-QS	Mean	1132.00	2746.00	1612.00	47484.00	38816.00	**146812.00**	**73936.00**	**126688.00**	400000.00	68692.00
	St.D.	345.51	1153.64	1122.43	6969.52	12083.42	**9138.53**	**15295.25**	**5281.69**	0.00	13109.38

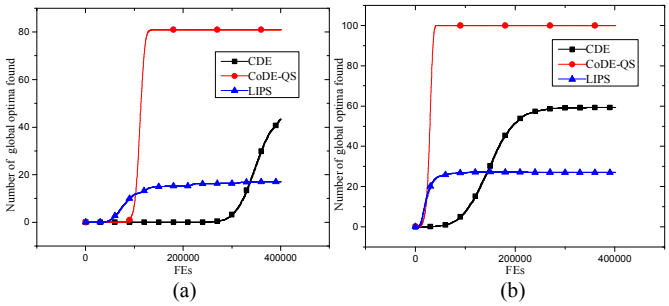


Fig. 5. Number of global optima found versus FEs. (a) F_6(3D) (b) F_7(3D)

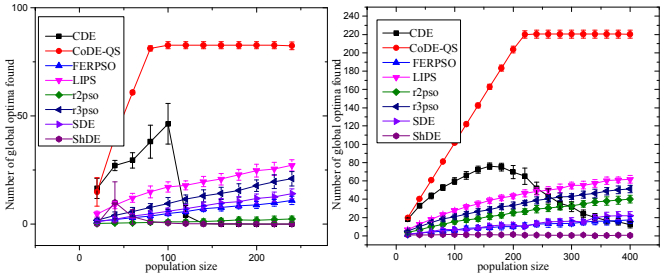


Fig. 6. Number of global optima found using different population sizes. (a) F_6(3D) (b) F_7(3D)

achieved by picking top-ranking individuals from different species. QS is integrated with composite differential evolution (CoDE) to tackle multimodal problems. The resulting algorithm (called CoDE-QS) inherits the global search ability of CoDE and is capable of maintaining multiple good solutions. Experiments on CEC2013 test suite have been carried out to investigate the performance of CoDE-QS. Experimental results show that CoDE-QS is very competitive. For future research, it would be interesting to integrate QS to other EAs for multimodal optimization. In addition, like most niching EAs, CoDE-QS has its niching parameter, the niche radius. An important issue is to develop methods to adaptively determine the niching parameter.

REFERENCES

[1] T. Back, 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] 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.

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

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

[10] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Apr. 2009.

[11] J. Brest, S. Greiner, B. Boskovi ˇ c, M. Mernik, and V. ´ Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[12] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, Feb. 2011.

[13] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[14] R. Storn and K. V. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, 1995

[15] X. Li, "Efficient differential evolution using speciation for multimodal function optimization," in *Proc. Conf. Genet. Evol. Comput.*, 2005, pp. 873–880

[16] J. Ronkkonen and J. Lampinen, "On determining multiple global optima by differential evolution," in *Proc. Eurogen Evol. Deterministic Methods Design Optimization Control*, Jun. 2007, pp. 146–151.

[17] Z. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinuous spaces," in *Proc. 15th Midwest Artif. Intell. Cognit. Sci. Conf*., Apr. 2004, pp. 92–97.

[18] 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.

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

[20] X. Li, "A multimodal particle swarm optimizer based on fitness Euclidean-distance ration," in *Proc. Genet. Evol. Computat. Conf*., 2007, pp. 78–85

[21] 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

[22] 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.

[23] S. Elsayed, R. Saker, and D. Essam, "An improved self-adaptive differential evolution algorithm for optimization problems," *IEEE Trans. Ind. Inf.*, vol. 9, no. 1, pp. 89–99, Feb. 2012.

[24] R. Saker, S. Elsayed, and T. Ray, "Differential evolution with dynamic parameters selection for optimization problems", *IEEE Trans. Evol. Computat*., vol. 18, no. 5, pp. 689–707, Sep. 2013.

[25] K. V. Price, "An introduction to differential evolution," in New Ideas Optimization. London, U.K.: McGraw-Hill, 1999, pp. 293–298

[26] 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.