Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2016) 24: 621 – 638

doi:10.3906/elk-1307-184

Solving multimodal optimization problems based on efficient partitioning of genotypic search space

Atabak Mashhadi KASHTIBAN[1,*], Sohrab KHANMOHAMMADI[2]

[1]Department of Electrical Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran
[2]Department of Control Engineering, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

Received: 23.07.2013 • **Accepted/Published Online:** 23.12.2013 • **Final Version:** 05.02.2016

Abstract: Enhancing the exploration and exploitation of multimodal optimization is still an interesting and challenging problem in optimization. Here we present a new population-based evolutionary algorithm for multiple optimal determinations. The approach is simply based on partitioning the genotypic search space and running a simple genetic algorithm with a small population within each partition. To increase the efficiency of the algorithm, the first-order discrete derivative of the fitness function for elite solutions was used to omit extra solutions. If the derivative of the fitness function is larger than a specified gradient threshold, no optimum exists within a given partition, and no population is created there after the first iteration. Except the adjusted gradient threshold, the proposed algorithm requires no other parameters rather than those of classical genetic algorithms. Moreover, unlike the other well-known algorithms, the proposed algorithm is not sensitive to the niche radius, and it needs no prior knowledge about the fitness function. The algorithm was tested for 10 multimodal benchmark functions, and its results were compared with the other related algorithms considering 4 commonly performance criteria. Our results show that the proposed algorithm is not only acceptable in terms of diversification and function evaluation number, but also has improved efficiency as compared to the others.

Key words: Evolutionary algorithms, multimodal optimization problem, genotypic search space partitioning

1. Introduction

In real-world problems such as planning, design and operational research problems, control engineering, and optimal control, finding all multiple extrema of the objective function is desirable. For practical engineering problems, due to time and cost constraints, robustness and stability considerations, and system performance enhancement like transitions to the optimal operational point in online systems, all alternative solutions are required to be found. In the optimization literature, multimodal optimization (MMO) refers to the art of finding the number and location of all the global and local optima of an objective function in the search space. In light of their high performance, efficiency, and ability to capture multiple solutions in a single run, evolutionary algorithms (EAs) and especially genetic algorithms (GA) have been widely used for solving MMO problems in the last decade.

Many techniques, such as the niching method, have been developed in recent years for preserving diversity in the fitness landscape and promoting convergence of the GA. Nearly two decades ago original versions of niching methods were proposed, including crowding [1], where offsprings are compared with their parents or a random subpopulation and then replaced with the most similar individuals; fitness sharing [2], which penalized and

*Correspondence: akeshtiban@gmail.com

degraded the fitness of an individual with respect to the presence of neighboring individuals within the niche radius; clustering [3], which divides the population into some clusters on the basis of calculated k-means distances between individuals and tries to locate all the optima at the center of the clusters; restricted tournament selection [4], which selects offsprings with random individuals having minimum hamming or Euclidian distance; clearing [5], which shares all of the resources among best individuals within the clearing radius (winners) and eliminates or decreases other individuals' fitness; and species conservation [6], which, according to individuals' similarity, divides the population into species and transitions individuals worth preserving within the species radius, i.e. the dominant individuals, into the next generation.

The aforementioned methods fail to fully preserve all the optima in the search space and maintain the diversity of the algorithm. Some of the problems with these methods along with proposed solutions are as follows. First, to set the niche radius in the sharing and cleaning methods, prior knowledge about the fitness landing is required; this problem has been tackled by dynamic niche clustering [7], sharing with population analysis considering varying cluster size [3], the adaptive sequential niche technique [8,9], and crowding clustering [10]. Second, tuning the selection pressure in very rugged functions having both smooth and sharp optima (such as Rastrigin's and Shubert's functions) is difficult, especially in the niching method, which can be overcome with the use of island models [11]. Third, some methods such as sharing, particularly when the dimension of the objective function or number of optima is high, are computationally expensive, resulting in the reduction of the efficiency of the algorithms. Sequential fitness sharing [8] and dynamic niche sharing [12] have been proposed for overcoming this weakness. Fourth, diversity maintenance in methods like clustering, crowding, and clearing, which are based on keeping super individuals (elites), is difficult. To establish a balance between elitism and diversity, deterministic and probabilistic crowding [13,14], a species-conserving GA [15], an adaptive elitist population-based GA [16], and an adaptive elitist-population strategy [17] have been put forward. Fifth, in niche-based algorithms, many parameters are required to be set. To remove these adjustable parameters, there have been attempts to design parallel or series GAs to divide the population into many subpopulations according to each optimum [18–21]. In new versions of these parallel methods, several populations associated with different niche methods share their best individuals for the next generation via special selection rules [22].

In view of its easy implementation and robust adaptability, the particle swarm optimization (PSO) method has recently been increasingly used for solving MMO problems; it can, in combination with the niching method, other EAs, or local searches, improve the algorithm efficiency [9,23–29]. Furthermore, several combinations of other artificial intelligence (AI) methods with themselves or with classical optimization methods have been developed to increase the efficiency of the algorithm. Some of the popular synthesis or intelligence methods that maintain the diversity of MMO algorithms are memetic PSO [30], the fuzzy clustering-based niching approach [31], hybrid PSO-BFGS [32], the firefly algorithm [33], artificial immune systems [34,35], evolution strategy [36], differential evolution [37–39], niching-based ant colony optimization [40], and multiobjective optimization methods [41].

Here we introduce a simple iterative population-based algorithm, with improved precision and convergence, for solving MMO problems. In this algorithm, the genotypic search space is partitioned, and a simple GA searches the optima in each subregion using traditional GA operators. The residual individuals, especially at the boundaries of considered subregions, are eliminated based on the concept of the discrete derivative concept of the objective function. The proposed algorithm requires no prior knowledge about the fitness landscape and only the GA parameters are needed to be set; moreover, no additional EA scheme such as the external

population for elite individuals is used in the algorithm. To evaluate the performance of the proposed method and compare the obtained results with other reports, we have used 10 difficult benchmark functions with 4 performance criteria.

The rest of the paper is organized as follows: in Section 2, after reviewing the related previous works, the proposed algorithm is discussed in detail. Benchmark test function characteristics and performance measurements of the algorithm are given in Section 3. Section 4 is devoted to the experimental and simulation results using 10 multimodal test functions and 4 performance criteria. Comparison results with other well-known algorithms are also presented in that section. Finally, the paper is concluded in Section 5.

2. Algorithm background and methodology

In this section, some population-based algorithms that divide the genotypic search space are reviewed. Thereafter, the fundamental concepts behind our proposed algorithm are elaborated.

2.1. Related works

One of the first attempts at solving MMO problems through dividing the genotypic or phenotypic search space, in the respectively so-called g-forking genetic algorithm (g-fGA) and p-forking genetic algorithm (p-fGA), was made by Tsutsui and Fujimoto [19]. In g-fGA all individuals in each subspace are forking to parent and child populations. Then, to obtain the salient schema (optima point) within each subspace, parent and child populations are respectively evolved through blocking and shrinking modes; then the best individuals in the child population are copied to the parent population. On the other hand, in the p-fGA, the search space is divided by defining a neighborhood hypercube around the best solutions in the phenotypic parameter space; then the child and parent individuals are respectively evolved inside and outside of the considered hypercube, with the use of the neighborhood hypercube size. The forking algorithm suffers from many drawbacks, including: 1) tuning many parameters such as the schema threshold, salient schema constant, neighborhood hypercube size, and population elitist selection; 2) dependency upon the chromosome length to calculate the neighborhood hypercube size and hamming distance; 3) the necessity of the binary GA for calculating the salient schema; and 4) the need for a large number of function evaluations (NFEs). Later on, Ursem [20] proposed a multinational evolutionary algorithm that groups the individuals, each covering one part of the fitness landscape. Grouping of individuals to other nations is done with the use of the hill-valley detection procedure. Based on the fitness of a random individual in a hill-valley, new nations are created and the individual immigrates to other nations, merging the same nations together; after some iteration, the capital of each government is an optimum point. If some part of the fitness landscape is very rugged and some other part is very smooth, the performance of the hill-valley procedure will diminish. Furthermore, to improve the diversity within each nation, the distance to policy, which plays the mutation role, should be chosen adaptively [42]. The roaming technique is another algorithm for solving MMO using subpopulation evolution. After the division of the search space, the best individuals in each subpopulation are detected, and the stability measure (the likeness to the optima point) of them is calculated. If the stability measure is larger than the stability threshold (individual is 1-stable), the considered chromosome is added to the archive as a candidate optima point; for a stability measure larger than the roaming subpopulation, the algorithm switches to another part of the search space using a high mutation rate. Besides using the external population, the algorithm suffers from the drawback that if the number of optima is very high or all of them have the same fitness, e.g., in Schaffer's, Sheckel's, and Roots's functions, emigration of roaming subpopulations to another point of the search space will not help increase the diversity of the algorithm.

2.2. Methodology of the proposed algorithm

The proposed method aims at finding all the global/local optima in multimodal problems with an acceptable compromise between exploration and exploitation of obtained solutions, minimum running time (function evaluations) of the algorithm, and considerable scalability with a low number of setting parameters. To this end, keeping the original genetic algorithm methodology and operators, we have made use of the concept of the discrete derivative of the fitness function instead of using hybrid or combinational AI methods.

Figure 1 shows the pseudocode of the proposed algorithm. First, the genotypic search space is divided into some Np equal partitions. Since we do not have any prior information about the number of optima and fitness function variation, the number of partitions is considered as a parameter of the algorithm, independent of the dimension of the fitness function or the number of optima. This partitioning of the genotype search space divides the fitness function with some simple curves, assuming that each optimum is located within only one partition. Thereafter, a random population chromosome is created within each partition with the size of the population being 5 to 20; all chromosomes within each partition are evolved by a simple GA with the same crossover selection operators and probability; and then the elite solutions of each partition are extracted. Figure 2 shows the distribution of the solutions over the fitness function after first iteration. As seen from the Figure 2, some of the solutions are located near actual optima and some at the highest border of the partition with better fitness. To eliminate these solutions, the first-order discrete derivative of the fitness function is calculated using Eq. (1) where ε is the gradient threshold, n is candidate optimum, and n – 1 and n + 1 are the previous and next closest chromosomes within the accuracy level. The absolute gradient of the fitness function near the actual optima is approximately zero, whereas the gradient of the solutions located at the border of the partitions is considerably high. Therefore, solutions except for those between partition c and d and within the partition g in Figure 2 all have a gradient larger than ε and are removed. To increase the exploration and exploitation of the algorithm, two policies are adopted.

$$\nabla f(n) \cong \min\{|f(n+1) - f(n)|, |f(n) - f(n-1)|\} \leq \varepsilon \tag{1}$$

1) After the first iteration, if the solution gradient is higher than the adjusted threshold gradient, no population is created within those partitions. Hence, the algorithm focuses on the partitions including the actual optima.

2) If two optima are located within the same partition, the number of partitions should be increased. In this case, two optima are too similar and hence their hamming distance in two consecutive iterations is smaller than the partition's length. Therefore, the number of partitions will be increased so that all the conditions are satisfied. The hamming distance is defined only for strings of the same length and denotes the number of places in which two strings have different characters.

According to the proposed method, the population size within each partition, GA operators such as crossover and selection, gradient threshold, and crossover probability will be kept constant across generations. The algorithm does not require a large population, an external population to save elite solutions, or any previous information on the fitness function and it has no parameter to be set. Therefore, the search space of the GA is limited to each partition. Note that the population size within each partition is small and hence in all iterations the mutation rate is set to zero.

Input : Np- number of partition for sub genetic algorithms to run

: PopSize- populations size for each partitions

: N_{ite}- maximum number of iterations

: N_{gen}- maximum number of generations

: Pc- crossover probability

: ε - gradient threshold

Output: the set of all optima

Partition the genotype space to Np partition (P_1,P_2,…,P_{Np})

For each iteration from 1 to N_{ite}

{

For each partition from P_1 to P_{Np}

{

Initialize a population with the size of PopSize and each individual containing L binary genes.

Execute a simple GA for N_{gen} generation with the crossover probability of Pc. Save the current evolved population in an array.

Calculate the gradient of the elite individual for the current partition, and save it into the current population, and remove extra individuals in the border and extra partitions.

Calculate the hamming distance of the elite individual for current partition into the current population and create a new partition if necessary.

}

}

Calculate the MPR, execution time, number of function evaluation and standard deviation.

Figure 1. The pseudocode of the proposed algorithm.

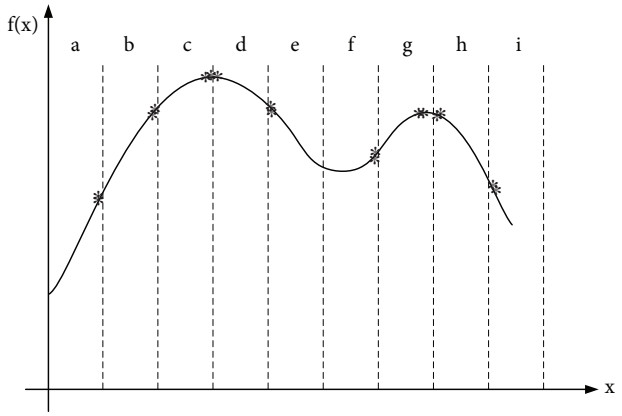


Figure 2. A schematic illustration of the solutions' distribution over the fitness function after the first iteration. All the solutions located at the border of two partitions will be removed according to the gradient condition of the algorithm.

3. Evaluation of the proposed algorithm

Here we evaluate the performance of the proposed algorithm after introducing 10 well-known benchmark functions [43]. After applying the proposed algorithm, 4 standard performance metrics of the obtained results were calculated. The numerical findings have been compared with other similar related works.

3.1. Test functions

To test and analyze the proposed method, we consider a set of unconstrained multimodal benchmark functions with different characteristics. Table 1 shows the multimodal test functions, their initial intervals, and the number of global/local optima for maximization problem. Some of these functions have symmetric landscapes or equal optima (e.g., F1, F5, F9), some have unequal optima (e.g., F2, F3) with a varying optima distribution over the landscape (e.g., F4, F7, F10), and some functions have both rigid and very smooth fitness landscapes (e.g., F4, F7, F9). Figure 3 shows the 1-dimensional (1D) plot of F_{3-5} and 2-dimensional (2D) surface plot of F_{8-10}. The specifications and characteristics of the considered test functions are briefly reviewed here. F1 (Deb's function), which has 5 equal peaks, is defined over the interval [0, 1]. F2 (Deb's function), defined over [0, 1], has 1 global maximum and 4 local maxima. F3 (Beasley's function), defined over [0, 1], has 5 decreasing optima. F4 (1D decreasing Shekel function) has 5 peaks (1 global and 4 local maxima) with different widths and sharpnesses. Traditional niche methods such as sharing and crowding could not solve this problem and their parameters (niche radius and cluster distance) need to be chosen very adaptively [44]. F5 has 9 uneven optima with different heights and spans over the interval [0, 1]. F6 (1D Shekel with 4 variables) is the 1D version of the Shekel function, dramatically fluctuating so that all optima are spread unevenly through the fitness landscape. F7 (Michalewicz) has a single global maximum, 1 global maximum, and an infinite number of local maxima located at the orthogonal lines. Traditional niche methods have much difficulty in finding all the optima for this function. F8 (six-hump camel back) is a 3D function with 2 global maxima, 2 stable local maxima, and 2 unstable local maxima being symmetric about the origin. F9 (Ursem f3) is a symmetric 2D function with 15 optima located at either very smooth or very sharp regions. F10 (waves), an asymmetric function, has peaks on boundary and flat hills. Some optimum regions are wide and vast, and some of them are malformed.

3.2. Performance measurements

As discussed earlier, the final aim of the MMO is to find all optima in the search space with acceptable diversity and convergence speed. To analyze the performance of the proposed algorithm, 4 criteria are used to measure the performance of the algorithm, including:

1) Success rate (SR), the percentage of independent runs for which all obtained solutions are located near local/global optima. That is, the success rate is the percentage of runs with their solutions located at an acceptable distance (the level of accuracy) of the actual solutions.

2) Maximum peak ratio (MPR), the ratio of the obtained solutions' summation to the actual solutions' summation, is defined as in Eq. (2) [36], where q is the number of the actual optima, and f_i and F_i are the fitness values of the obtained and actual optima within the final generation, respectively. The MPR indicates the quality of the obtained results and does not give any information about the diversity of the results.

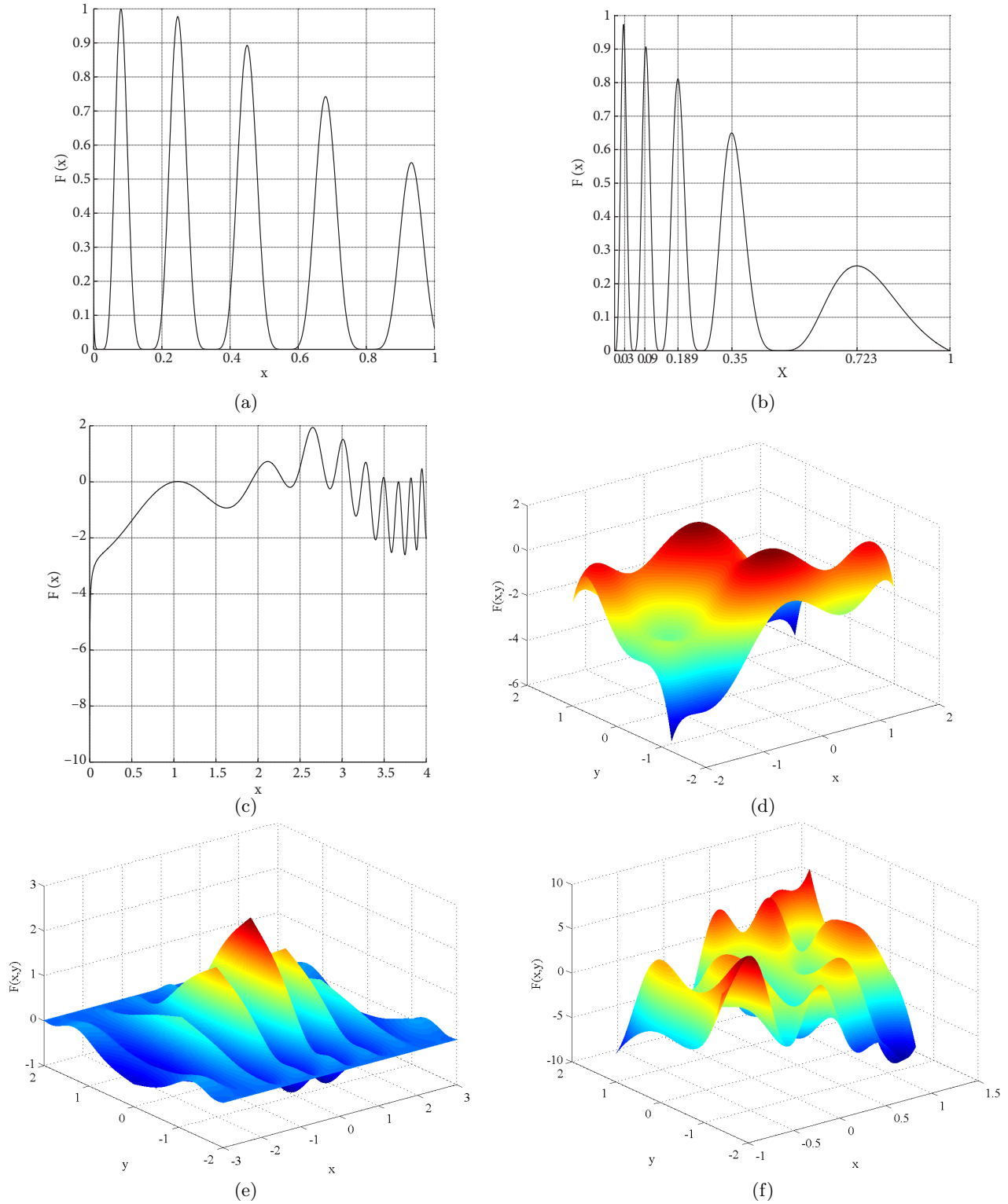


Figure 3. Illustrations of multimodal functions F_{3-5} and F_{8-10} in one and two dimensions. All functions have uneven and asymmetric optima that spread in both rigid and very smooth fitness landscapes. According to Table 1, all objective functions are plotted versus x and y parameters. a) Beasley's function (F3), b) 1D decreasing Shekel function (F4), c) uneven optima function (F5), d) 2D plot of six-hump camel back function (F8), e) 2D plot of Ursem f3 function (F9), f) 2D plot of waves function (F10).

Table 1. Test functions, initialization interval, and their number of global/local optima.

Name	Test function	Range	Global/ local
F1	$f_1(x) = \sin^6(5\pi x)$	$0 \le x \le 1$	5/0
F2	$f_2(x) = \exp\left[-2\log(2) \times \left(\frac{x-0.1}{0.8}\right)^2\right] \times \sin^6(5\pi x)$	$0 \le x \le 1$	1/4
F3	$f_3(x) = \exp\left[-2\log(2) \times \left(\frac{x-0.08}{0.854}\right)^2\right] \times \sin^6\left(5\pi\left(x^{0.75} - 0.05\right)\right)$	$0 \le x \le 1$	1/4
F4	$f_4(x) = (1-x) \times \sin^4\left(\frac{5\pi}{(1+x)^4}\right)$	$0 \le x \le 1$	1/4
F5	$f_5(x) = \ln(x)\left(\sin\left(e^x\right) + \sin(3x)\right)$	$0 \le x \le 4$	1/8
F6	$f_6(x) = \sum_{i=1}^{10}\left[(x-a_i)^T(x-a_i) + c_i\right]^{-1}$ where i | a_i^T | | | | c_i 1 | 4.0 | 4.0 | 4.0 | 4.0 | 0.1 2 | 1.0 | 1.0 | 1.0 | 1.0 | 0.2 3 | 8.0 | 8.0 | 8.0 | 8.0 | 0.2 4 | 6.0 | 6.0 | 6.0 | 6.0 | 0.4 5 | 3.0 | 7.0 | 3.0 | 7.0 | 0.4 6 | 2.0 | 9.0 | 2.0 | 9.0 | 0.6 7 | 5.0 | 5.0 | 3.0 | 3.0 | 0.3 8 | 8.0 | 1.0 | 8.0 | 1.0 | 0.7 9 | 6.0 | 2.0 | 6.0 | 2.0 | 0.5 10 | 7.0 | 3.6 | 7.0 | 3.6 | 0.5	$0 \le x \le 10$	1/7
F7	$f_7(x,y) = \sin(x)\sin^{20}\left(\frac{x^2}{\pi}\right) + \sin(y)\sin^{20}\left(\frac{2y^2}{\pi}\right)$	$0 \le x, y \le \pi$	$1/many$
F8	$f_8(x,y) = -\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + \left(-4 + 4y^2\right)y^2\right]$	$-1.9 \le x \le 1.9$ $-1.1 \le y \le 1.1$	1/5
F9	$f_9(x,y) = \sin(2.2\pi x + 0.5\pi)\frac{2-\lvert y\rvert}{2}\frac{3-\lvert x\rvert}{2} + \sin\left(0.5\pi y^2 + 0.5\pi\right)\frac{2-\lvert y\rvert}{2}\frac{2-\lvert x\rvert}{2}$	$-3 \le x \le 3$ $-2 \le y \le 2$	2/14
F10	$f_{10}(x,y) = (0.3x)^3 - \left(x^2 - 4.5y^2\right)xy - 4.7\cos\left(3x - y^2(2+x)\right)\sin(2.5\pi x)$	$-0.9 \le x \le 1.2$ $-1.2 \le y \le 1.2$	1/9

$$MPR = \frac{\sum_{i=1}^{q} f_i}{\sum_{i=1}^{q} F_i} \tag{2}$$

3) The chi-square like deviation (P_{chi}), a measure defined as in Eq. (3) [44], indicates the quality of the obtained results' distribution over the actual local/global values within the level of accuracy. A low P_{chi} indicates that the obtained results tend to be very close to real optima. Obviously, P_{chi} is reduced over generations. In Eq. (3), μ_i and σ_i are the mean and standard deviation of n_i solutions, respectively, over the accuracy level of the ith optima in the last iteration, and q is the number of the actual optima.

$$\begin{aligned} P_{chi} &= \sqrt{\sum_{i=1}^{q+1}\left(\frac{n_i-\mu_i}{\sigma_i}\right)^2} \\ \sigma_i &= \sqrt{\frac{1}{N}\sum_{i=1}^{q+1}(n_i-\mu_i)^2} \end{aligned} \tag{3}$$

4) The box plot, a nonparametric graphical method to display differences within a data set without making any assumptions about the statistical distribution. It summarizes the degree of data dispersion with five numbers: the minimum and maximum of data, the lower and upper quartile, and the median of the data. That is, the box plot of the last generation's solutions show how close the median number is to the actual optima and how other obtained results are distributed around the median number within the accuracy level.

4. Experiments and results

The simulation was implemented using the Microsoft Visual C# compiler in a computer with Intel Core 2 Duo CPU running at 2.0 GHz with 2 GB of RAM and a Vista Home premium operating system. Table 2 shows the dimension of the objective function, population size within each partition, number of generations, level of accuracy, gradient threshold, and number of partitions for each objective function. As explained earlier, the number of partitions is a parameter of the algorithm. To prevent losing any optima, a high number of partitions was selected; that is, it was chosen to be respectively 10 and 20 for 1D and 2D objective functions. The crossover operator was chosen to be roulette wheel selection, and the crossover probability and chromosome length were respectively set to 0.9 and 20 for all functions. The algorithm was run 50 times (iteration) for each test function, and the best fitted optimal solutions were obtained.

Table 2. The population size within each partition, number of generations, level of accuracy, gradient threshold, and partition number for each objective function.

Function	Dimension	Population size	Number of generation	Level of accuracy	Gradient threshold (ε)	Initial partition count
F1	2D	5	5	0.000001	0.001	10
F2	2D	5	5	0.00001	0.001	10
F3	2D	10	20	0.00001	0.001	10
F4	2D	10	20	0.00001	0.001	10
F5	2D	10	20	0.05	0.001	10
F6	2D	5	5	0.00005	0.001	10
F7	3D	10	10	0.05	0.05	20
F8	3D	5	5	0.005	0.05	20
F9	3D	10	10	0.005	0.05	20
F10	3D	10	20	0.05	0.05	20

4.1. Success rate and MPR

Table 3 presents the execution time, the success rate, and the MPR of the obtained results within the accuracy level and number of function evaluations. For more challenging problems (F7 and F9), although the MPR is relatively low, the value of the SR is significantly near 1.00. This happens because both functions have sharp and smooth peaks simultaneously. This problem could be solved more efficiently if the gradient threshold were varied adaptively in each optimum point. As seen from Table 3, the proposed algorithm not only achieves all the global/local maxima with a good performance in terms of success rate and MPR (more efficient) but can also balance between the number of function evaluations and execution time (more effective).

Figures 4 and 5 give a clearer view of the performance of the proposed algorithm. Figure 4 shows the distribution of the obtained solutions over the fitness landscape of F1–F6, and Figure 5 shows the contour plot

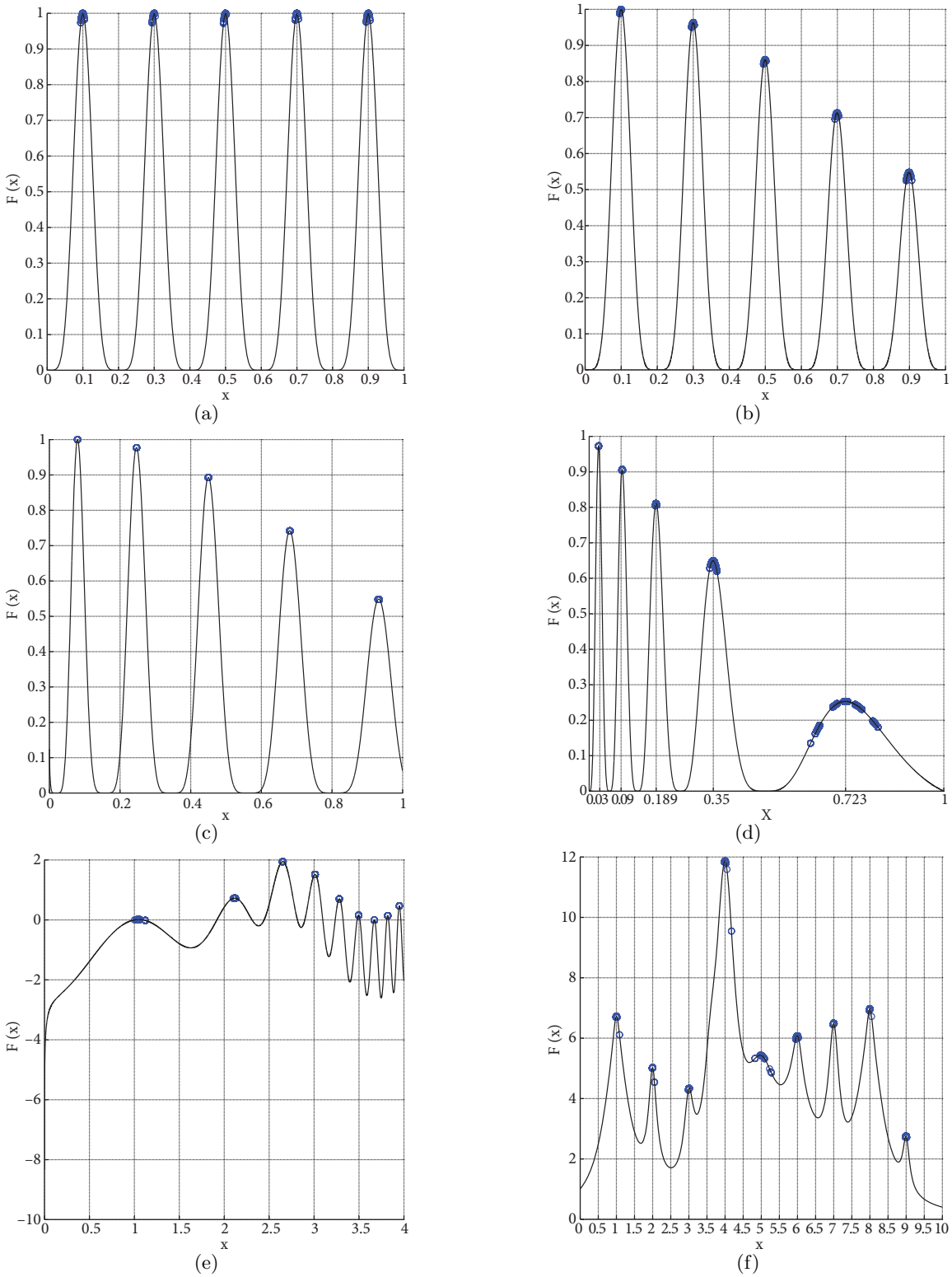
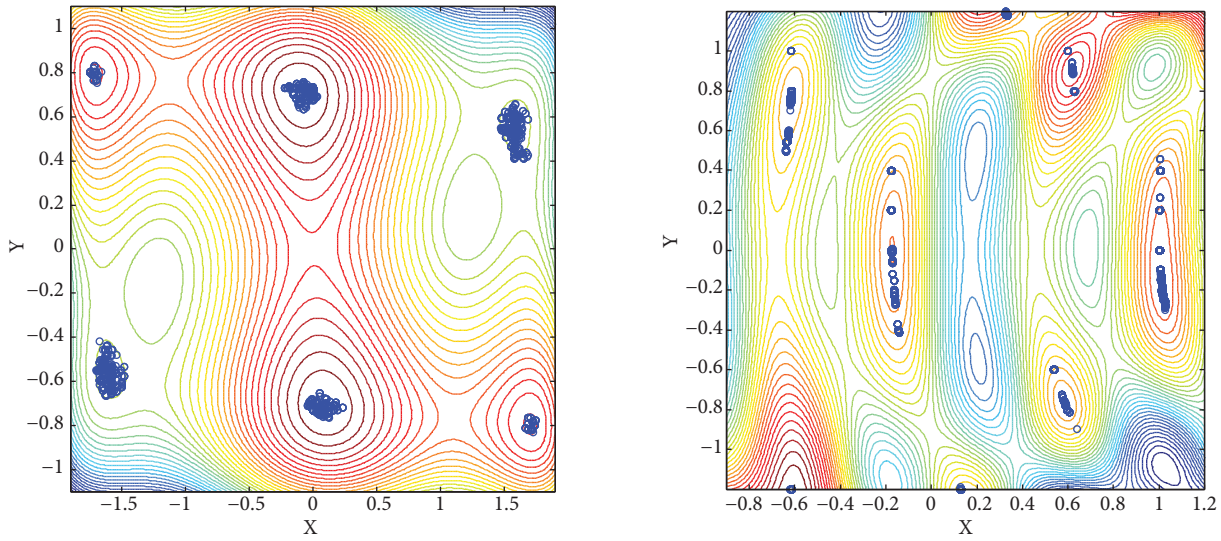


Figure 4. Snapshots of the results distribution over the fitness landscape of F1–F6 in the last iteration for 50 runs. Horizontal axis (x) shows the genotypic value of optima points. a) Deb's function (F1), b) Deb's function (F2), c) Beasley's function (F3), d) 1D decreasing Shekel function (F4), e) uneven optima function (F5), f) 1D Shekel function with 4 variables (F6).

Table 3. The execution time, success rate, maximum peak ratio, standard deviation, P_{chi}, and number of function evaluations for 50 runs.

Function number	Time (s)	SR	MPR	Standard deviation	P_{chi}	NFEs
F1	0.12	1.00	1.00	0.00120	4.82	470
F2	0.14	1.00	0.9998	0.0099	4.63	470
F3	2.38	0.98	0.9967	0.0051	6.1	10,300
F4	3.17	0.98	0.9950	0.0065	6.14	21,400
F5	5.10	1.00	0.9879	0.0132	5.86	35,375
F6	0.35	1.00	0.9995	0.0374	4.46	1100
F7	4.54	0.98	0.9493	0.0282	8.31	24,640
F8	1.33	1.00	0.9940	0.0271	9.86	7080
F9	4.52	1.00	0.9560	0.0212	7.10	24,640
F10	11.52	0.98	0.9875	0.1265	8.33	66,000

of the solution distribution for F8 and F9; as seen, the proposed algorithm has properly located all the local and global optima.



(a) (b)

Figure 5. Snapshots of the results' distribution over the counter plot landscape of F_8 and F_9 in the last iteration for 50 runs. Horizontal and vertical axes (x, y) show the genotypic value of optima points, whereas the color of counter plot lines represents the height of the optima points. a) Counter plot of six-hump camel back function (F8), b) counter plot of Ursem f3 function (F9).

4.2. Standard deviation and P_{chi}

Besides the success rate and the MPR, the standard deviation and P_{chi} of optima found are also among the most important criteria to show the quality of the results' distribution. The value of the standard deviation is dependent on the accuracy level; indeed, with a very low accuracy level, we can reduce the standard deviation. The numerical simulations show that the population size within each partition does not have any significant

Table 4. Comparisons results with differential evolution, PSO niching, and its related algorithms.

Function		This work	FER-PSO-LS	SPSO-LS	R3PSO-LS	R3PSO-LHC	MSPO	Crowding DE-STL
F1	SR	1.00	1.00	1.00	1.00	0.992	1.00	0.846
	MPR	1.00	0.999	0.999	0.999	0.999	-	-
F2	SR	1.00	0.24	1.00	0.04	0.16	1.00	-
	MPR	0.9998	0.5229	1.00	0.3498	0.6299	-	-
F3	SR	0.993	0.00	0.00	0.00	0.00	1.00	1.00
	MPR	0.9967	0.288	0.288	0.4411	0.8718	-	-
F8	SR	1.00	0.76	0.52	0.64	0.72	-	-
	MPR	0.9940	1.00	0.38	1.00	1.00	-	-

Table 5. Success rate comparison results with other numerical optimization algorithms.

Test function		This work	BMPGA	MPGA	DNC	MNGA	Clearing
F5	SR	1.00	1.00	1.00	1.00	1.00	0.967
F8	SR	1.00	0.933	0.733	0.033	0.067	0.067
F10	SR	0.98	1.00	0.933	0.067	0.964	1.00

impact on the performance of the algorithm. Table 3 shows the execution time, the standard deviation of the obtained results within the accuracy level, and P_{chi}. The high value of the standard deviation in F10 is because of its irregular behavior and infinite local optima. F5 also has the same situation, as some of the optima are located at sharp regions and some at smooth regions.

4.3. Box plot

As mentioned, the box plot provides a revealing summary of the data distribution around actual optima. Figure 6 shows the distribution of solutions around the actual optima in the box plot format for F_1–F_6. The horizontal lines within the boxes show the median, while the upper and lower ends of the boxes are the upper and lower quartiles. The dashed appendages illustrate the spread and shape of the distribution, and the dots represent the outlying values. Clearly, the length between the lower and upper quartiles is very small, and the median number is very close to the actual optima. The boxes of the same functions are not comparatively tall or short with respect to each other. This shows that within the considered accuracy level, the distribution of obtained results around actual optima is relatively uniform.

4.4. Effect of different partition counts and gradient threshold

In this section, we compare the performance of the proposed algorithm with different partition counts and different gradient thresholds (ε) on the F5 and F8 functions. As discussed in Section 3.1, the optima points of F5 are different in width and sharpness and some optima points of F8 are unstable. Therefore, their performance is very sensitive to good tuning of partition counts and gradient threshold. We perform the experiments using partition count values of 5, 10, 15, 20, 25, and 30, whereas other algorithm parameters values are in accordance with Table 2. Figures 7a–7d show the obtained results of the SR, MPR, P_{chi}, and time execution of algorithm for the F5 and F8 functions with varying partition counts. From Figures 7b and 7d, as expected, as the partition counts are increased, time execution of the algorithm is increased significantly, whereas the distribution of obtained results around actual optima does not decrease. According to performance-increasing policies that

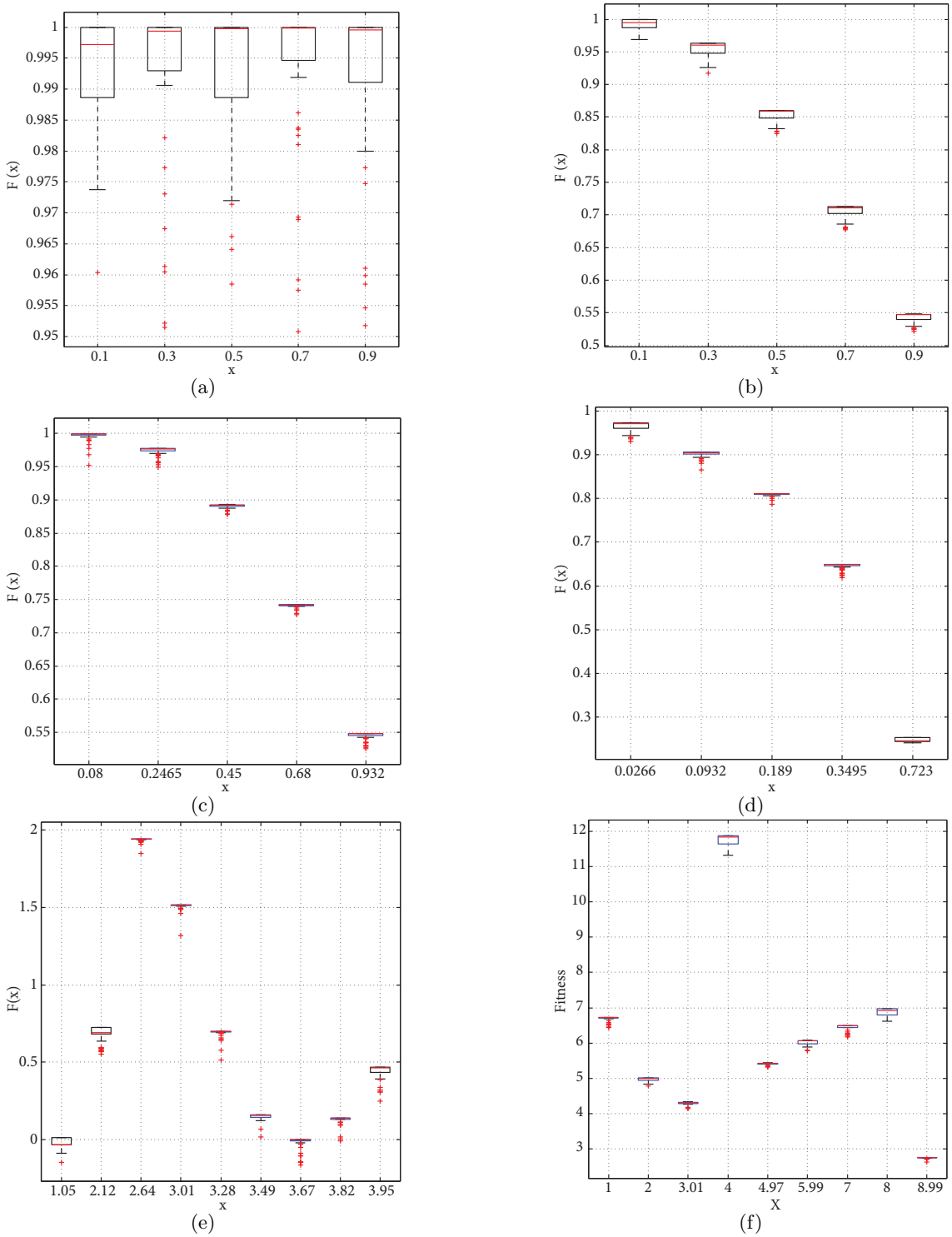


Figure 6. Box plots of results achieved by 50 runs for F1–F6 functions. The box has lines at the lower quartile, median, and upper quartile values. The whiskers extending to the most extreme data points not considered outliers. Outliers (denoted by +) are data with values beyond 100 units of interquartile range. a) Box plots of results for Deb's functions (F1), b) box plots of results for Deb's function (F2), c) box plots of results for Beasley's function (F3), d) box plots of results for 1D decreasing Shekel function (F4), e) box plots of results for uneven optima function (F5), f) box plots for 1D Shekel function with 4 variables (F6).

were introduced in Section 2.2, notice that after a certain number for partition counts (10 for F5 and 20 for F8), SR and MPR get significantly closer to 1.00. In other words, the proposed algorithm is relatively robust against the population count varying.

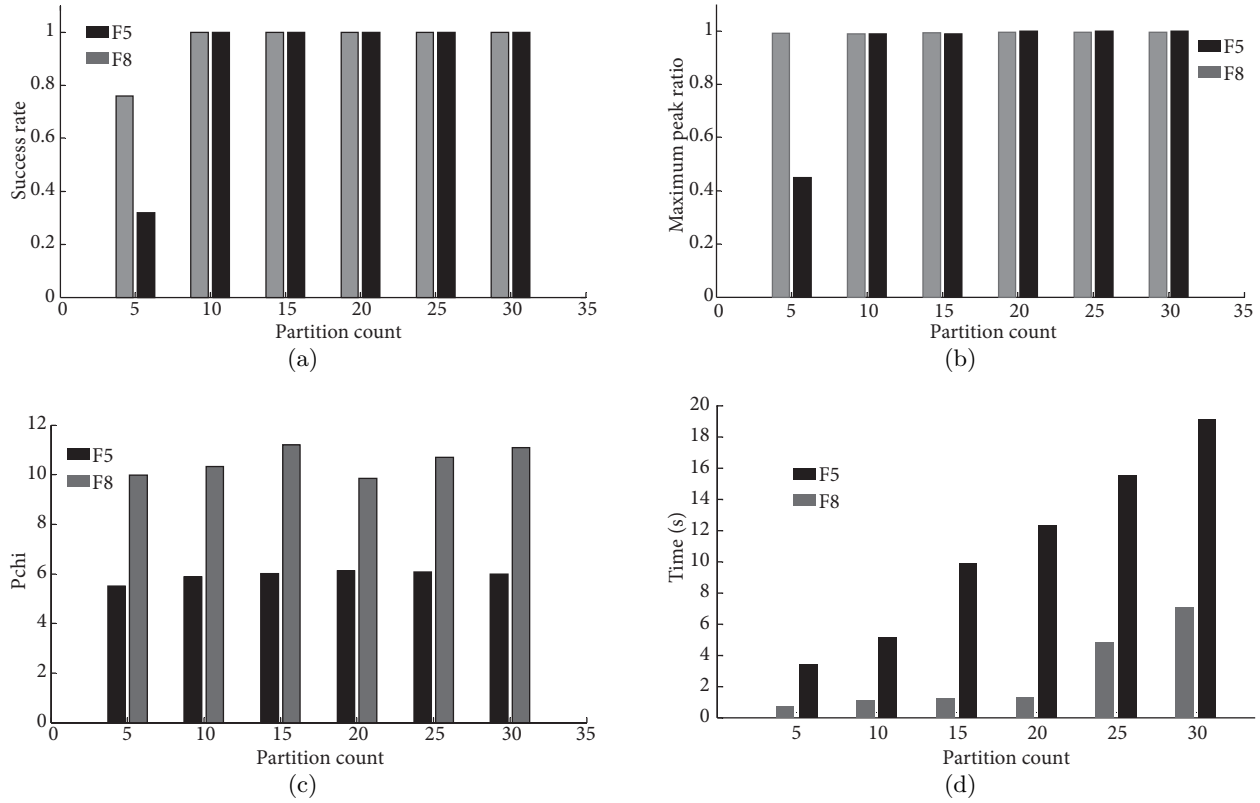


Figure 7. The comparisons of success rate (SR), maximum peak ratio (MPR), P_{chi}, and execution time values for F5 and F8 functions when partition counts are varying for 50 runs. Horizontal axis represents the partition counts values. a) SR of F5 and F8 functions with varying partition counts value, b) MPR of F5 and F8 functions with varying partition counts value, c) P_{chi} of F5 and F8 functions with varying partition counts value, d) execution time of F5 and F8 functions with varying partition counts value.

To examine the influence of the gradient threshold, ε, we conduct a series of experiments, again using the F5 and F8 functions, in which we set the value of ε to 1E-04, 1E-03, 1E-02, 5E-02, and 1E-01 and record the SR, MPR, P_{chi}, and time execution of the algorithm whereas other algorithm parameters values are in accordance with Table 2. Figures 8a–8d show the obtained results of the SR, MPR, P_{chi}, and time execution of algorithm for the F5 and F8 functions with varying gradient thresholds. It is obvious that when the gradient threshold decreases, the computational effort of the algorithm for finding the actual optima within considered accuracy level is increased. It causes the execution time of the algorithm to grow significantly. From Figure 8d, as the gradient threshold is increased, time execution of the algorithm increases significantly while P_{chi}, SR, and MPR changes are not considerable. In other words, for better performance, a good compromise between ε and time execution is necessary. From Figure 8, we notice that the best final results for ε are 1E-03 and 5E-02 for the F5 and F8 functions, respectively.

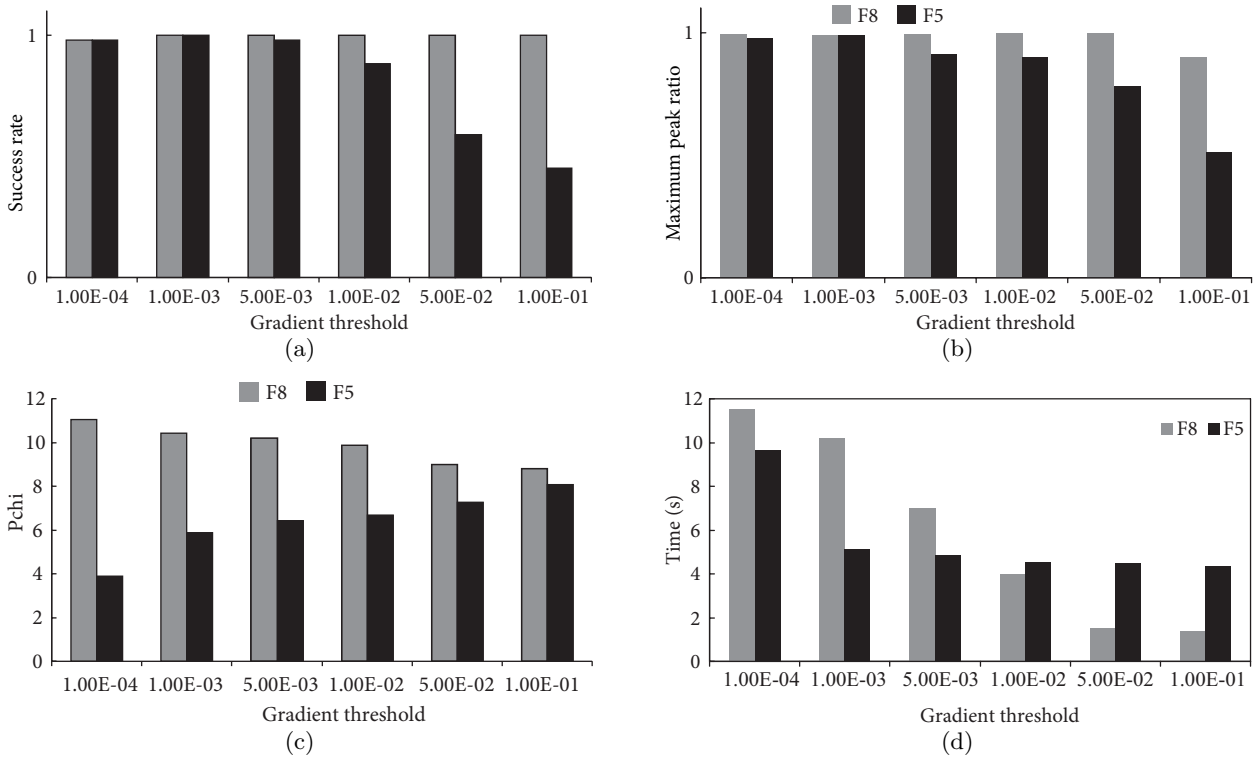


Figure 8. The comparisons of success rate (SR), maximum peak ratio (MPR), P_{chi}, and execution time values for F5 and F8 functions when gradient threshold varies for 50 runs. Horizontal axis represents the value of gradient threshold. a) SR of F5 and F8 functions with varying gradient threshold value, b) MPR of F5 and F8 functions with varying gradient threshold value, c) Pchi of F5 and F8 functions with varying gradient threshold gradient value, d) execution time of F5 and F8 functions with varying gradient threshold value.

4.5. Comparison with other niching algorithms

The proposed algorithm is compared with a number of well-known multimodal methods:

1. FER-PSO-LS [43]: fitness Euclidian distance ratio PSO with local search
2. SPSO [43]: speciation-based PSO with local search
3. R3PSO-LS [43]: PSO with ring topology and local search
4. R3PSO-LHC [43]: PSO with ring topology and local search and hill climbing
5. MSPO [30]: adaptive local search with PSO
6. Crowding DE-STL [45]: crowding differential evolution using spatial and temporal locality
7. DE/isolated /1 [46]: differential evolution based on the selection isolated vector
8. BMPGA [41]: biobjective multipopulation genetic algorithm
9. MPGA [41]: multipopulation genetic algorithm
10. MNGA [41]: multinational GA
11. DNC [41]: dynamic niche clustering
12. Clearing methods [41]

In Tables 4 and 5 the SR and MPR of different multimodal algorithms are compared. Algorithms 1–4 are executed on a computer with Intel Pentium 4 CPU, 2.50 GHz, and 2 GB of memory with 25 runs and a maximum of 10,000 NFEs using MATLAB 7.1. Algorithm 5 is executed with 30 runs and a maximum of 30,000 NFEs. Algorithm 6 is implemented on the EC4 framework using Sun's Java programming language with 50 runs and a maximum of 10,000 NFEs, and the results of Algorithm 7 is obtained with 100 runs. Algorithms 8–12 are implemented on a computer with Intel Xeon 2.66 GHz with 512 KB of cache and 512 MB DDR RAM, running the Red Hat Linux 8.0.3.2-7 platform with 30 runs and more than 100,000 NFEs. The dashes in Table 4 indicate that the corresponding results are not available from the literature. In contrast to other well-known algorithms (especially PSO niching methods and related algorithms), the proposed method performed better in terms of accuracy (success rate and MPR).

5. Conclusion

Here a novel population-based evolutionary algorithm for multimodal function optimization was put forward. The algorithm partitioned the genotypic search space so that each would be ideally associated with a peak of the fitness function. Within each partition, a simple GA with a small population was evolved. To increase the efficiency and speed of the algorithm, the first-order discrete derivative of elite solutions was used to remove extra solutions, particularly those located on the border of two partitions. If the derivative of the fitness function is larger than a specified value (the gradient threshold), there are no optima within the considered partition, and consequently no population is created within this region after the first iteration. Therefore, the number of partitions is optimized. Except for the adjusted gradient threshold, the proposed algorithm does not need the niche radius or any prior knowledge about the objective function and many parameters settings. The proposed algorithm was experimentally tested with a difficult test suite consisting of 10 benchmark multimodal function optimizations. Considering 4 performance measures, the performance of the proposed algorithm was calculated and compared against existing well-known algorithms. The experimental results show that the proposed method is very efficient and superior in accuracy and number of function evaluations when compared to existing well-known algorithms.

For future work, it is straightforward to determine the number of clusters with mathematical methods such as multivariate statistical methods or other intelligent methods such as self-organizing neural networks. Another interesting research work is to combine the proposed method with local search methods or other evolutionary algorithms such as differential evolution. It would also be interesting to use the gradient threshold adaptively or examine the performance of the proposed algorithm with other criteria such as the Mann–Whitney U test and the two-sample Kolmogorov–Smirnov test.

References

[1] De Jong KA. An analysis of the behavior of a class of genetic adaptive systems. PhD, University of Michigan, Ann Arbor, MI, USA, 1975.

[2] Goldberg DE, Richardson J. Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the Second International Conference on Genetic Algorithms; 1987. pp. 41–49.

[3] Yin X, Germay N. A fast genetic algorithm with sharing scheme using cluster analysis methods in multi-modal function optimization. In: Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms; 1993. pp. 450–457.

[4] Harik G. Finding multi-modal solutions using restricted tournament selection. In: Proceedings of the Sixth International Conference on Genetic Algorithms; 1997. pp. 24–31.

[5] Pétrowski A. A clearing procedure as a niching method for genetic algorithms. In: Proceedings of Third IEEE International Conference on Evolutionary Computation; 1996. pp. 798–803.

[6] Li JP, Balazs ME, Parks GT, Clarkson PJ. A species conserving genetic algorithm for multi-modal function optimization. IEEE T Evol Comput 2002; 10: 207–234.

[7] Gan J, Warwick K. Dynamic niche clustering: a fuzzy variable radius niching technique for multimodal optimization in gas. In: Proceedings of IEEE Congress on Evolutionary Computation; 2001. pp. 215–222.

[8] Beasley D, Bull DR, Matin RR. A sequential niche technique for multimodal function optimization. IEEE T Evol Comput 1993; 1: 101–125.

[9] Zhang J, Huang DS, Lok TM, Lyu MR. A novel adaptive sequential niche technique for multimodal function optimization. Neurocomputing 2006; 69: 2396–2401.

[10] Qing L, Gang W, Zaiyue Y, Qiuping W. Crowding clustering genetic algorithm for multimodal function optimization. Appl Soft Comput 2008; 8: 88–95.

[11] Siarry P, Petrowski A, Bessaou M. A multipopulation genetic algorithm aimed at multimodal optimization. Adv Eng Softw 2002; 33: 207–213.

[12] Miller BL, Shaw MJ. Genetic algorithms with dynamic niche sharing for multimodal function optimization. In: Proceedings of the IEEE International Conference on Evolutionary Computation; 1996. pp. 786–791.

[13] Mahfoud SW. Crowding and Preselection Revisited. Technical report, IlliGAL. Urbana-Champaign, IL, USA: University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory, 1992.

[14] Mengshoel OJ, Goldberg DE. The crowding approach to niching in genetic algorithms. IEEE T Evolut Comput 2008; 16: 315–354.

[15] Li JP, Balazs ME, Parks GT, Glarkson PJ. A species conserving genetic algorithms for multimodal function optimization. IEEE T Evolut Comput 2002; 10: 207–234.

[16] Leung KS, Liang Y. Adaptive elitist-population based genetic algorithm for multimodal function optimization. In: Proceedings of Genetic and Evolutionary Computation—GECCO; 2003. pp. 1160–1171.

[17] Liang Y, Leung KS. Genetic algorithm with adaptive elitist-population strategies for multimodal function optimization. Appl Soft Comput 2011; 11: 2017–2034.

[18] Elo S. A parallel genetic algorithm on the CM-2 for multi-modal optimization. In: Proceedings of International Conference on Evolutionary Computation; 1994. pp. 818–822.

[19] Tsutsui S, Fujimoto Y. Forking genetic algorithms: GAs with search space division schemes. IEEE T Evolut Comput 1997; 5: 61–80.

[20] Ursem RK. Multinational evolutionary algorithms. In: Proceedings of International Conference on Evolutionary Computation; 1999. pp. 1633–1640.

[21] Lung RL. A subpopulation stability based evolutionary technique for multimodal optimization. In: Proceeding of GECCO; 2004. pp. 26–30.

[22] Yu EL, Suganthan PN. An ensemble of niching algorithms. Inform Sciences 2010; 180: 2815–2833.

[23] Parsopoulos KE, Vrahatis MN. On the computation of all global minimizers through particle swarm optimization. IEEE T Evolut Comput 2004; 8: 211–224.

[24] Parrot D, Li X. Locating and tracking multiple dynamic optimal by a particle swarm model using speciation. IEEE T Evolut Comput 2006; 10: 440–458.

[25] Vitela JE, Castanos O. A sequential niching memetic algorithm for continuous multimodal function optimization. Appl Math Comput 2012; 218: 8242–8259.

[26] Juang YT, Tung SL, Chiu HC. Adaptive fuzzy particle swarm optimization for global optimization of multimodal functions. Inform Sciences 2011; 181: 4539–4549.

[27] Qu BY, Suganthan PN, Das S. A distance-based locally informed particle swarm model for multi-modal optimization. IEEE T Evolut Comput 2012; 17: 387–402.

[28] Krohling RA, Mendel E, Campos M. Swarm algorithms with chaotic jumps for optimization of multimodal functions. Eng Optimiz 2011; 43: 1243–1261.

[29] Fan SS, Liang YC, Zahara E. Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. Eng Optimiz 2004; 36: 401–418.

[30] Wang H, Moon I, Yang S, Wang D. A memetic particle swarm optimization algorithm for multimodal optimization problems. Inform Sciences 2012; 197: 38–52.

[31] Imrani AE, Bouroumi A, Abidine ZE, Limouri M, Essaid A. A fuzzy clustering-based niching approach to multimodal function optimization. Cogn Syst Res 2000; 1: 119–133.

[32] Li S, Tan M, Tsang W, Kwok JTY. A hybrid PSO-BFGS Strategy for global optimization of multimodal functions. IEEE T Syst Man Cyb 2011; 41: 1003–1014.

[33] Yang XS. Firefly algorithms for multimodal optimization. In: Proceedings of SAGA'09; 2009. pp. 169–178.

[34] Yap DFW, Koh SP, Tiong SK, Prajindra SK. A swarm-based artificial immune system for solving multimodal functions. Appl Artif Intell 2011; 25: 693–707.

[35] Woldermariam KM, Yen GG. Vaccine-enhanced artificial immune system for multimodal function optimization. IEEE T Syst Man Cyb 2010; 40: 218–228.

[36] Shir OM, Emmerich M, Back T. Adaptive niche radii and niche shapes approaches for niching with the CMA-ES. IEEE T Evolut Comput 2010; 18: 97–126.

[37] Thomsen R. Multimodal optimization using crowding-based differential evolution. In: Proceedings of the Congress on Evolutionary Computation; 2004. pp. 1382–1389.

[38] Hendershot ZV. A differential evolution algorithm for automatically discovering multiple global optima in multi-dimensional discontinuous spaces. In: Proceedings of the Artificial Intelligence and Cognitive Sciences; 2004. pp. 92–97.

[39] Zaharie D. Extensions of differential evolution algorithms for multimodal optimization. In: Proceedings of SYNASC; 2004. pp. 523–534.

[40] Angus D. Niching for ant colony optimization. In: Lewis A, Mostaghim S, Randall M, editors. Biologically-Inspired Optimisation Methods. Heidelberg, Germany: Springer, 2009. pp. 165–188.

[41] Yao J, Kharma N, Grogono P. Bi-objective multipopulation genetic algorithm for multimodal function optimization. IEEE T Evolut Comput 2010; 14: 80–102.

[42] Ursem RK. Multinational gas: multimodal optimization techniques in dynamic environments. In: Proceedings of GECCO; 2000. pp. 1633–1640.

[43] Qu BY, Liang JJ, Suganthan PN. Niching particle swarm optimization with local search for multi-modal optimization. Inform Sciences 2012; 197: 131–143.

[44] Lin CY, Wu WH. Niche identification techniques in multimodal genetic search with sharing scheme. Adv Eng Softw 2002; 33: 779–791.

[45] Wong KC, Wu CH, Mok RKP, Peng C, Zhang Z. Evolutionary multimodal optimization using the principle of locality. Inform Sciences 2012; 194: 138–170.

[46] Otani T, Suzuki R, Arita T. DE/isolated/1: A new mutation operator for multimodal optimization with differential evolution. International Journal of Machine Learning and Cybernetics 2013; 4: 99–105.