Comparison of Multi-Modal Optimization Algorithms Based on Evolutionary Algorithms

Gulshan Singh[*]
Computational Design Optimization Center
Wright State University, Dayton, Ohio
singh.31@wright.edu

Dr. Kalyanmoy Deb [†]
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, India
deb@iitk.ac.in

ABSTRACT

Many engineering optimization tasks involve finding more than one optimum solution. The present study provides a comprehensive review of the existing work done in the field of multi-modal function optimization and provides a critical analysis of the existing methods. Existing niching methods are analyzed and an improved niching method is proposed. To achieve this purpose, we first give an introduction to niching and diversity preservation, followed by discussion of a number of algorithms. Thereafter, a comparison of clearing, clustering, deterministic crowding, probabilistic crowding, restricted tournament selection, sharing, species conserving genetic algorithms is made. A modified niching-based technique – modified clearing approach – is introduced and also compared with existing methods. For comparison, a versatile hump test function is also proposed and used together with two other functions. The ability of the algorithms in finding, locating, and maintaining multiple optima is judged using two performance measures: (i) number of peaks maintained, and (ii) computational time. Based on the results, we conclude that the restricted tournament selection and the proposed modified clearing approaches are better in terms of finding and maintaining the multiple optima.

Categories and Subject Descriptors

G.1 [**Numerical analysis**]: OptimizationUnconstrained optimization; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms

[*]Graduate Research Assistant, Wright State University
[†]Professor, Indian Institute of Technology Kanpur.

Keywords

Niching, Clearing, Crowding, Restricted Tournament Selection, Sharing, Species Conserving GA.

1. INTRODUCTION TO MULTI-MODAL OPTIMIZATION

As the name suggests, multi-modal functions have multiple optimum solutions, of which many are local optimal solutions. Multi-modality in a search and optimization problem, usually causes difficulty to any optimization algorithm in terms of finding the global optimum solutions. This is because in these problems there exist many attractors for which finding a global optimum can become a challenge to any optimization algorithm. An example of multi-modal function having 20 maxima with unequal peak heights is shown in Figure 1. In the case of peaks of equal value (height), the convergence to every peak is desirable, whereas with peaks of unequal value, in addition to knowing the best solutions, one may also be interested in knowing other optimal solutions. Therefore, when dealing with multi-modal functions, some modification is necessary to the standard GAs to permit stable subpopulations at all peaks in the search space.

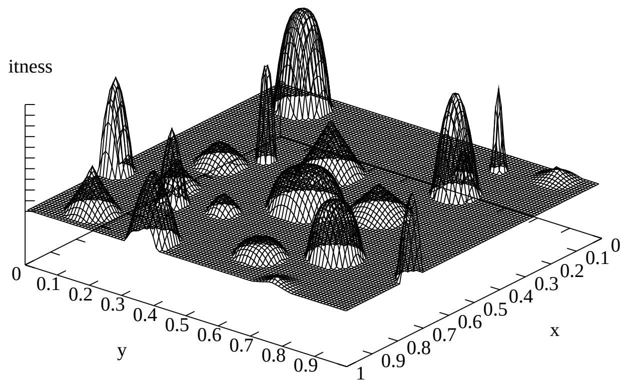


Figure 1: A multi-modal function.

It is clear from the above discussion that the task in a multi-modal optimization problem is to find multiple optima (global and local). If a classical point-by-point approach is used for this purpose, the method must have to be used many times, every time hoping to find one of the optima. However, evolutionary algorithms (EAs) have an niche over their classical counterparts in solving multi-modal optimization problems. Instead of applying an optimization method again and again, an EA's population approach can be exploited so that it can be applied only one time to find and maintain multiple optimal solutions.

Multi-modal EAs dates back to the ground-breaking work of Goldberg and Richardson [6], in which they nicely showed how a niche-preserving technique can be introduced in a standard genetic algorithm and multiple optimal solutions can be obtained. Since that study, many researchers have suggested methodologies of introducing niche-preserving techniques so that, for each optimum solution, a niche gets formed in the population in an evolutionary algorithm. In this paper, we collect seven such main strategies and provide a brief description of each of them. We also suggest an extension of an existing clearing strategy. Despite suggestion of such varied methodologies, there does not seem to exist any study which systematically compared the methodologies. Here, we make an attempt and apply all eight approaches to two existing five-modal problems. In addition, we have suggested a versatile hump test problem which is scalable to any number of variables, number and nature of optimal basins in the search space. Algorithms are compared on hump problems having as many as 25 variables and 50 optima. Based on the repeated simulation results, conclusions about the relative performance of the niching methodologies is discussed.

2. EXISTING NICHING METHODS

Niching methods have been developed to reduce the effect of genetic drift resulting from the selection operator in the standard GAs. They maintain diversity in the population and permit the GAs to find many optima in parallel. A niche is commonly referred to as an optimum of the domain and the fitness represents the resources of that niche.

A niching method must be able to form and maintain multiple, diverse, final solutions, whether these solutions are of identical fitness or different fitnesses. A niching method must be able to maintain these solutions for a large enough iterations. It is well known that a reduction in selection pressure, selection noise, and operator disruption does not typically result in a niching GA. What is required is not just a slow selection process, nor a selection with reduced noise, nor less disruptive operators, but a new type of algorithm – one that promotes diversity along useful dimensions of diversity, while allowing other dimensions to converge. A niching method alters the selection operator to provide selection pressure within, but not across regions of the search space.

In the following subsections, we present various multi-modal evolutionary algorithms available in the literature. The symbols used for various niching methods considered for comparison in this paper are as follows:

A1	: Clearing
A2	: Clustering
A3	: Deterministic Crowding
A4	: Probabilistic Crowding
A5	: Restricted Tournament Selection
A6	: Sharing
A7	: Species Conserving GA
A8	: Modified Clearing (proposed in this study)

2.1 Clearing

Petrowski [15], [16], [17] suggested the clearing procedure, which is a niching method inspired by the principle of sharing of limited resources within subpopulations of individuals characterized by some similarities. Instead of evenly sharing the available resources among the individuals of a subpopulation, the clearing procedure supplies these resources only to the best individual of each subpopulation. The clearing is naturally adapted to elitist strategies.

The clearing procedure is applied after evaluating the fitness of individuals and before applying the selection operator. The population is sorted from best to worst according to the fitness values. Thereafter, all solutions having a critical distance measure ($\sigma_{\rm clear}$) from the best κ solutions in the population are *cleared*, meaning that their fitness values are set to zero. However, the fitness of the best κ solutions are kept as they are. After the clearing is over, the solutions closer to the next best κ solutions are cleared as before and the fitnesses of these next best κ solutions are not changed. This procedure is continued till all solutions are considered. The clearing procedure was shown to efficiently reduce the genetic drift when used with an appropriate selection operator.

2.2 Clustering

Yin [19], [20] proposed a niching scheme using a clustering methodology. A clustering algorithm is used to divide the population into niches. The fitness is calculated based on the distance d_{ic} between the individual and its niche centroid. This significantly reduces the time complexity. The final fitness of an individual is calculated by the relation:

$$F_i = \frac{f_i}{n_c\left(1-(d_{ic}/2d_{\max})^{\alpha}\right)}, \qquad (1)$$

where n_c is the number of individuals in the niche containing the the individual i, $d_{\max}$ is the maximum distance allowed between an individual and its niche centroid, and α is a constant. The formation of the niches is based on the adaptive Macqueen's K-mean algorithm. The algorithm begins with a fixed number (k) of seed points taken as the best k individuals. Using a minimum allowable distance $d_{\min}$ between niche centroids, a few clusters are formed from the seed points. The remaining population members are then added to these existing clusters or are used to form new clusters based on $d_{\min}$ and $d_{\max}$. These computations are performed in each generation. As long as the number of clusters is not $O(N)$, this algorithm is computationally faster than the sharing function approach.

2.3 Crowding: Restricted Replacement

Crowding, originally proposed by De Jong [1], is motivated by analogy with competition for limited resources among similar members of a natural population. Dissimilar population members, often of differing species, occupying different environmental niches, and therefore do not typically compete for resources. Similar individuals on the other hand, tend to occupy the same environmental niches, and must compete for the same limited resources. When a niche has reached its carrying capacity, weaker members of that niche will be crowded out of the population by stronger members. Older members of niche will eventually be replaced by fittest of the younger members. In De Jong's [1] crowding mechanism newly created individuals in a population replace similar individuals. His method followed the standard genetic algorithms except that only a fraction of the population reproduces and dies in each generation and a percentage of the population, specified by the *generation gap* (G), is chosen via fitness proportionate selection to undergo crossover and mutation. A random sample of CF individuals is taken from the population, where CF is called *crowding factor*. Of the CF elements, the one most similar to the element being inserted gets replaced. Similarity is defined using phenotype distance (distance in the decision parameter space) matching.

2.3.1 *Deterministic Crowding*

Mahfoud in [13], [11], [12] analyzed De Jong's [1] crowding factor model and attributed its inability to maintain more than two peaks of a multi-modal landscape due to stochastic errors in replacement which create genetic drift and fixation. Subsequent modifications to the algorithm included elimination of parameter requirements, reduction of replacement errors and restoration of selection pressure. This leads to a new crowding algorithm *deterministic crowding* [13], [11], [12] that was capable of maintaining multiple peaks.

In the deterministic crowding, the population is randomly paired into $N/2$ pairs, where N is population size, of individuals to undergo crossover to yield two offspring, which then compete with parents for replacement. The pair of tournament that force the closest competition is held, closeness being computed as shown below on the basis of average distance between the pairs of parent-child on the basis of phenotypic similarity. In case of a tie, parents are preferred. The following procedure is to be performed $N/2$ times and the overall procedure is to be repeated g generations:

1. Select two parents, p_1 and p_2, randomly with no replacement
2. Perform a crossover between them, yielding c_1 and c_2
3. Apply mutation/other operators, yielding c'_1 and c'_2
4. If $[d(p_1, c'_1) + d(p_2, c'_2) \leq d(p_1, c'_2) + d(p_2, c'_1)]$
   - If $f(c'_1) \geq f(p_1)$ replace p_1 with c'_1
   - If $f(c'_2) \geq f(p_2)$ replace p_2 with c'_2

   else
   - If $f(c'_2) \geq f(p_1)$ replace p_1 with c'_2
   - If $f(c'_1) \geq f(p_2)$ replace p_2 with c'_1

2.3.2 *Probabilistic Crowding*

Most of the crowding algorithms encountered until now, replaced the upfront selection pressure with selection pressure at the replacement stage through some form of localized tournaments between similar individuals. Since a deterministic tournament was used, such methods will always prefer higher fitness individuals over lower fitness individuals. This finally leads to a loss of niches, whenever the tournaments between global and local niches are played.

To prevent this deterministic nature of the algorithm and thus provide a restorative pressure in such cases, Mengshoel [14] proposed probabilistic crowding. In this case, a probabilistic acceptance (replacement) rule was proposed that permitted higher fitness individuals to win over lower fitness individuals in proportion to their fitness. This allows a restorative pressure and prevents the loss of niches of lower fitness. Essentially the algorithm is deterministic crowding with a probabilistic replacement operator.

In the probabilistic crowding, two similar individuals X and Y compete in a probabilistic tournament where the probability of X winning the tournament is given by:

$$p(X) = \frac{f(X)}{f(X) + f(Y)}, \tag{2}$$

where f is the fitness function.

2.4 Restricted Tournament Selection

Harik [8], [9] introduced a modified tournament algorithm that exhibited niching capabilities. Restricted tournament selection works, by initially selects two elements at random, A and B, from the population and perform crossover and mutation on these two elements resulting in two new elements, A' and B'. A' and B' are then to be placed into the population as in a steady state GAs. RTS scheme allows the GAs to choose which present element each inserted pair of element will replace. For each of A' and B' RTS scans w (*window size* analogous to CF in Crowding) more members of the population and picks the individual that most closely resemble A' or B' from those w elements. Let these elements be called A'' and B'' . A' then compete with A'' and if A' wins, it is allowed to enter the population. A similar competition occurs between $B'$and B''. This kind of tournament will restrict an entering element of the population from competing with others that are too different from it.

Harik tested his method on several multi-modal problems having the number of peaks varying from 5 to 32. The algorithm was able to maintain individuals at all the peaks, through some of the peaks gradually lost a number of elements. It was able to maintain all the global optima in the massively multi-modal problem.

2.5 Sharing

Goldberg and Richardson's [6] *fitness sharing* algorithm was one of the first attempt to deal directly with the location and preservation of multiple solution in a GAs. The fitness sharing algorithm took a cue from nature, restricting the multiple growth of one type of individuals by making each individual in the population to share its fitness assignment with nearby elements in the population. Genetic algorithms with sharing are well known for tackling multi-modal function optimization problems. Niches have been introduced in GAs by dividing the population into different subpopulation according to the similarity of the individuals. When a certain limiting number of individuals are occupying a niche, it becomes favorable for other individuals to search for a new niche available in the search space. Thus, an algorithm must find an equilibrium between the number of individuals occupying a niche and the payoff the niche. Such methods would lead to a state where the number of individuals occupying a niche is proportional to the fitness of the niche. It is implemented by degrading an individual's payoff due to the presence of other individuals in its neighborhood, and the amount of sharing contributed by each individual into its neighbor depends on the proximity between the two.

$$f_{\text{shared}}(i) = \frac{f_{\text{original}}(i)}{n_i} \qquad \text{where} \qquad n_i = \sum_{j=1}^{N} Sh(d_{ij}) \tag{3}$$

Here, $Sh(d_{ij})$ represents the sharing function which is a power function of the form:

$$Sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^{\alpha} & \text{if } d < \sigma_{share}; \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Other sharing functions can also be developed, through the above mentioned is by far the most prevalent. Here α denotes the scaling factor and σ_{share} the niche radius. The fitness-sharing algorithm require both a distance metric over the population space and parameter σshare. The σ_{share} parameter defines for each individual, the maximum distance over which it has to share its fitness with other population members. Later work on fitness-sharing by Deb and Goldberg [5] studied this parameter and involved a direct comparison between fitness-sharing and the crowding algorithm developed by De Jong [1]. Deb and Goldberg [4] showed that one possible way to set σ_{share} would be to divide the search space into a number of equal sized hyper-space equal to the number of sought out optima.

2.6 Species Conserving Genetic Algorithms

Species conserving genetic algorithms (SCGA) by Li et al. [10] is a recent technique for evolving parallel subpopulations using species conservation. SCGA is based on the concept of dividing the population into several species according to their similarity. Each of these species is built around a dominating individual called the species seed. Species seeds found in the current generation are saved (conserved) by moving them into the next generation. The proposers claim that the technique has been proven to be very effective in finding multiple solutions of multi-modal optimization problems.

The definition of a species, as well as the operation of the SCGA, depends on a parameter called the species distance, which denoted by σ_s. The species distance specifies the upper bound on the distance between two individuals for which they are considered to be similar. In their approach they proposed that the species distance be used to determine which individuals are worth preserving from one generation to the next.

A species is a subset, in which the distance between any two individuals is less than the species distance. Note that we do not require that any two individuals satisfying the condition that the distance between them is less than the species distance belong to the same species. In SCGA population is partitioned into species by a technique dominating individual or species seeds.

To determine the individuals that are to be copied into the next generation, we need to partition the current generation into a set of dominated species and determine the dominating individuals in each of these species. In the algorithm, X_s denotes the set of species found in generation t. The algorithm builds the set X_s by successively considering each of the individuals in t, in decreasing order of fitness. When an individual is considered, it is checked upon against the species seeds found so far. If X_s does not contain any seed that is closer than half the species distance ($\sigma_s/2$) to the individual considered, then the individual will be added to X_s.

Once all the species are been found, the new population is constructed by applying the usual genetic operators: selection, crossover, and mutation. Since some species may not survive following these operations, we copy them into the new population and thus enable them to survive. However, the number of species is always less than the population size.

3. PROPOSED MODIFIED CLEARING APPROACH

We modify the clearing approach discussed above, so as to make it more effective and reliable.

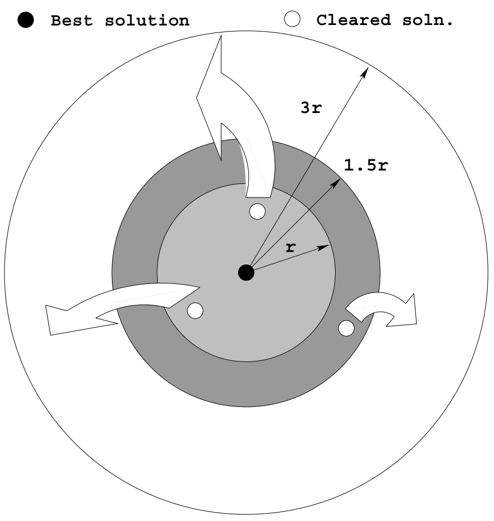


Figure 2: Shifting in individuals in modified clearing approach. The parameter r is the clearing radius.

Clearing takes the best individual (we choose $\kappa = 1$ here) and makes the fitness of all other individuals zero in the range of critical distance measure (σ_{clear}) except the best solution. This means that inferior individuals (cleared solutions) do not get a chance to participate in crossover and mutation operations. This is because they do not get selected due to having zero fitness value under a proportionate selection scheme. Although these solutions occupy population slots, these cleared solutions are of no use in the original clearing method. In the proposed modified clearing approach, instead of wasting population slots, we reallocate these individuals outside the range of their respective best individual in the hope of finding better interesting areas in the search space.

After doing the clearing, as described earlier, we search for the individuals which have a zero fitness value. Thereafter, each such solution is checked for its belonging to the basin of attraction of any solution having a non-zero fitness with 1.5 times the clearing radius or $1.5 \times \sigma_{\text{clear}}$. If any such solution is found, the zero-fitness solution is shifted to the region $1.5\times\sigma_{\text{clear}}$ to $3\times\sigma_{\text{clear}}$ at random, as shown in the Figure 2 and its fitness is evaluated again. This method has all strengths of the clearing method. In addition, the reallocation of bad solutions allows the method to explore the search space for better solutions. It is needless to write that such an modification comes with an extra computational cost of reallocating the bad solutions. Such a reallocation strategy can be repeated till a solution is placed outside the basin of all solutions having a non-zero fitness, the computational cost will be more. To simplify matters, we reallocate each solution only once and redo the clearing strategy.

4. COMPLEXITY OF VARIOUS NICHING METHODS

Table 1 shows the computational complexity of one iteration of various methods studied in this paper. The table shows that the crowding methods have the minimum complexity. For the sharing approach, the complexity is $O(N^2)$ which comes from calculating the niche count value. In the clearing approach, the complexity is smaller than $O(N^2)$, because a clearing step is applied only after sorting the individuals according to the their fitness values and not applied to every population member. The complexity for RTS is $N \times w$, where w is window size (the niching parameter used in RTS), since for every individual the distance is computed w times. The complexity of species converging approach is between $O(N)$ and $O(N^2)$. In the clustering algorithm, N_c is number of clusters and C is a constant, which is proportional to the number of iterations needed to find a stable set of clusters. The modified clearing has a reasonably large complexity, as the clearing is performed twice in every generation.

Table 1: Complexity of various methods used in this study.

	Algorithm	Complexity
1	Sharing	N^2
2	Clearing	CN
3	Deterministic Crowding	$O(N)$
4	Probabilistic Crowding	$O(N)$
5	RTS	$N \times w$
6	SCGA	$> O(N)$ and $< O(N^2)$
7	Clustering	$C \times N_c \times N$
8	Modified Clearing	$3 \times N^2$

5. TEST FUNCTIONS

Several test problems are considered to test and compare the niching schemes maintained above. The test functions are selected as representative of other functions with equal and unequal peaks and equal and unequal peak separation. The first two test problems were originally suggested elsewhere [4] and the third function is a multi-variable, multi-peaked function.

P1: A periodic function having peaks of equal size and interval:

$$P_1(x) = \sin^6(5\pi x). \qquad (5)$$

This function has five peaks in the interval $0 \le x \le 1$, as evident from the argument of the sine function and they are located at $x = 0.1$, 0.3, 0.5, 0.7 and 0.9. In all these locations, the objective function values are same and equal to one.

P2: A periodic function having peaks of unequal size and interval:

$$P_2(x) = e^{-2(\ln 2)(\frac{x-0.01}{0.8})^2} \sin^6(5\pi[x^{0.75} - 0.05]). \qquad (6)$$

This function is having five peaks function in the interval 0 $\le x \le 1$, with peaks of decreasing magnitude and increasing period.

P3: Hump Problem with arbitrary number of peaks:

Hump is a multi-variable function, in which K peaks (all maxima) are generated at random locations, with different shapes and size. All variables are initialized within $[0, 1]$. First, the location $\mathbf{x}_k^*$ and radius r_k of the basin of attraction of each maximum is randomly created so that the distance between the two neighboring maxima (l and m) is at least equal to $(r_l + r_m)$. For the k-th maxima, a peak-height h_k and a shape factor α_k are also randomly chosen. To compute the objective value of a solution $\mathbf{x}$, first the nearest peak (say k-th maximum, residing in the basin of attraction of k-th peak) is identified and then the Euclidean distance $d(\mathbf{x}, k)$ is identified between the solution $\mathbf{x}$ and the center of the k-th maximum. Then, the following equation is used to compute the function value:

$$f(\mathbf{x}) = \begin{cases} h_k \left[1 - \left(\frac{d(\mathbf{x},k)}{r_k}\right)^{\alpha_k}\right], & \text{if } d_{ik} \le r_k; \\ 0, & \text{otherwise.} \end{cases} \qquad (7)$$

With the above setting, multi-modal test problems having different complexities can be created by choosing h_k, r_k, α_k and maximum number of peaks K. Figure 1 shows the resulting hump function for a two-variable, 20-peaked ($K = 20$) problem having different values of h_k, r_k and α_k.

6. PERFORMANCE MEASURES

With the niching methods and functions chosen, performance measure is selected next to allow a suitable comparison of different methods. Since the number of optima to be found is a known quantity in these test problems, we use the number of obtained peaks N_{peaks} as a performance metric. Also, we use the computational time (on a Pentium IV 1.8 GHz) as the second performance measure.

7. SIMULATION RESULTS

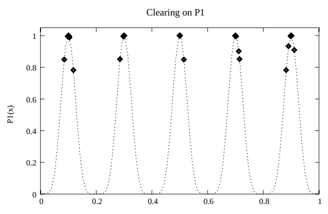
The niching methods A1-A8 are employed on the functions P1 and P2 with different parameter settings, which are shown in Table 2. Real-coded genetic algorithms with SBX [3] as a recombination operator, and the polynomial mutation operator [2] are used for all eight methods. Above parameters are found to perform (experimentally) the best for each of the algorithm and are held constant for all runs on functions P1 and P2. To minimize the stochastic error due to the selection procedure, the stochastic reminder selection method [7] is used.

7.1 Function P1

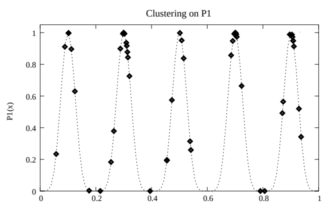
Function P1 has five peaks in the search space. Different niching methods (A1-A8) are applied on this function and solutions after 200 generations are plotted (and evaluated) on the function itself in Figure 3. Although solutions close to the true optima were found much earlier (around 100 generations), we run the algorithms for 200 generations to investigate if any algorithm had any effect of losing optima with generations. The convergence of population to all peaks can be clearly seen for clearing, deterministic crowding, RTS, and the modified clearing approaches. The distribution of solutions are the best for the RTS approach, followed by the clearing and the modified clearing approaches. The performance of probabilistic crowding and SCGA are worst and cannot find one of the five optima.

Table 2: Parameter settings for problems P1 and P2 used by all eight algorithms A1 to A8.

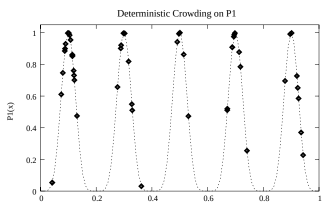
Algorithms	Pop.	Gen.	X-Over	Mu	η_m	η_c	Algorithm Specific parameter
A1	50	200	0.56	0.1	15	20	σ_{clear}=0.1, κ=1
A2	50	200	0.7	0.08	5	10	no of cluster=10,$d_{\min}$=0.04,d_{max}=0.1
A3	50	200	1.0	1.0	5	10	-
A4	50	200	1.0	1.0	5	10	-
A5	50	200	0.7	0.8	5	15	window size(w)=20
A6	50	200	0.8	0.08	15	20	σ_{share}=0.1,α=1.0
A7	50	200	0.9	0.05	5	10	species distance=0.01 & 0.02
A8	50	200	0.5	0.09	15	20	σ_{clear}=0.1, κ=1



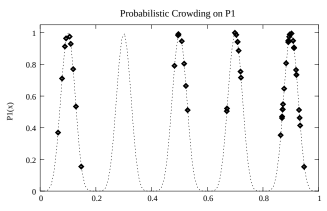
(a) Clearing



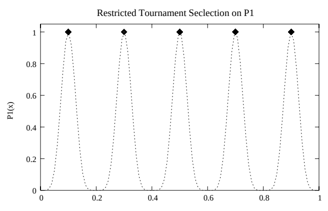
(b) Clustering



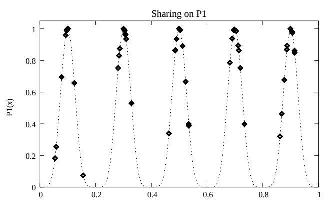
(c) Deterministic crowding



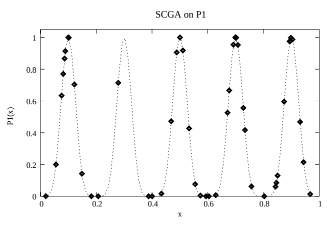
(d) Probabilistic crowding



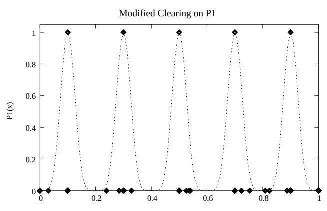
(e) RTS



(f) Sharing

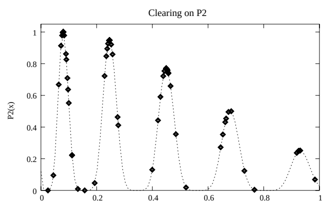


(g) SCGA

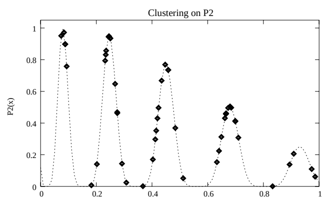


(h) Modified clearing

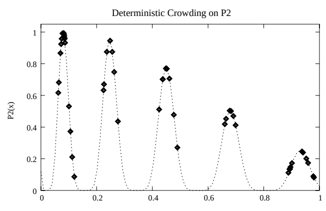
Figure 3: Function P1



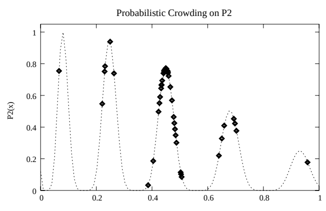
(a) Clearing



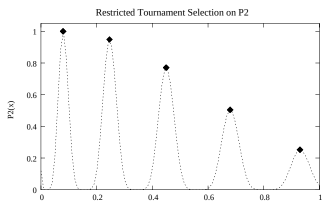
(b) Clustering



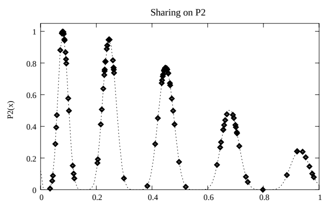
(c) Deterministic crowding



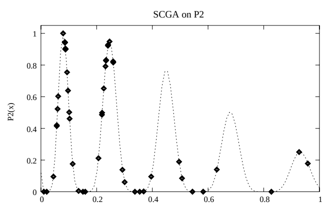
(d) Probabilistic crowding



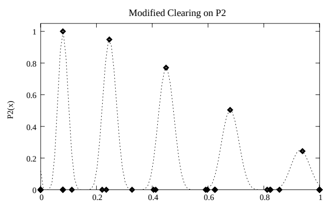
(e) RTS



(f) Sharing



(g) SCGA



(h) Modified clearing

Figure 4: Function P2.

7.2 Problem P2

Function P2 has five peaks and the peaks are unequally spaced and of unequal heights. Different niching methods (A1-A8) are applied on this function and solutions after 200 generations are plotted on the function itself. Figure 4 shows the obtained solutions by algorithms A1 to A8. RTS and the two clearing approaches perform best here. Although sharing approach finds all five peaks, the distribution of points around the optima is not crisp as it is in RTS or in the case of modified clearing. The effect of modified clearing over the original clearing approach is evident here.

From all above results, we can conclude that although all algorithms except the probabilistic crowding, species converging, and clustering approaches are able to find and maintain all optima in the above two problems, the restricted tournament selection, clearing, and the modified clearing approach find the crisply and with reliability. Similar conclusion about clearing method was also made in another study [18].

Table 3: Five-variable hump function.

K		A1	A2	A3	A4	A5	A6	A7	A8
20	SR	11.9	12.2	19.9	0.0	**20.0**	11.9	7.06	**20.0**
	SD	2.12	0.25	0.19	0.0	**0.00**	2.00	2.74	**0.00**
	TT	10.3	8.30	**3.78**	-.-	9.88	27.8	49.5	258.2
	TSD	0.19	2.45	0.34	-.-	1.33	0.75	36.4	66.3
30	SR	29.9	16.7	29.9	0.0	**30.0**	14.6	10.7	**30.0**
	SD	0.58	3.47	0.58	0.0	**0.00**	4.01	3.47	**0.00**
	TT	11.6	23.3	**6.08**	-.-	14.7	36.1	58.1	314.6
	TSD	1.52	15.8	**0.31**	-.-	4.27	0.50	55.5	104.5
40	SR	**40.0**	16.7	39.5	0.0	**40.0**	16.4	13.4	**40.0**
	SD	**0.00**	3.48	1.10	0.0	**0.00**	3.65	3.52	**0.00**
	TT	27.8	22.8	**8.54**	-.-	25.9	46.1	88.3	390.5
	TSD	1.29	17.2	**0.20**	-.-	0.50	1.14	32.7	52.8
50	SR	**50.0**	19.2	49.1	0.0	49.7	22.3	16.8	**50.0**
	SD	**0.00**	3.25	1.20	0.0	0.64	3.50	2.34	**0.00**
	TT	19.4	23.7	**11.2**	-.-	24.5	56.9	98.0	454.0
	TSD	0.96	8.83	**0.42**	-.-	1.33	1.30	75.4	257.0

7.3 Generic Hump Functions

The previous two problems involved only one variable. Here, we apply all eight multi-modal optimization algorithms on five to 25-variable hump function having as large as 50 optima. We have chosen a constant radius ($r_k = 0.29$ for five-variables, 0.60 for ten-variables, 1.45 for 25-variable problems), a constant height of $h_k = 1.0$, a constant shape parameter of $\alpha = 1.0$. An algorithm is said to have found a particular peak if it is able to find at least one solution within 0.15 times the radius of the peak from the midpoint of the peak. In the results presented here, tables show the success rate (average number of optima found over 50 independent runs), standard deviation in the success rate, computational time taken by an algorithm for a given number of overall evaluations and standard deviation in the computational time. In the tables, SR stands for success rate, SD is deviation in the success rate, TT is time taken, and TSD is the standard deviation in the time taken.

For each problem, ten different parameter settings are used to develop ten instantiations of the problem. Thereafter, each problem is solved with five different initial populations, therefore making a total of 50 runs for each algorithm and for each problem. Every hump function is constructed once and the same is used for all algorithms.

7.3.1 Five-Variable Hump Problem

First, we discuss the five-variable problem. The population size for 20, 30, 40 and 50-peak problem are chosen to be 800, 900, 1000 and 1100, respectively. The performance of all algorithms is shown in Table 3. From the table it is evident that the restricted tournament selection, deterministic crowding, clearing, and the modified clearing approaches perform better. In terms of computational time, the deterministic crowding is still much ahead of others. Just like the two-variable hump function, the modified clearing approach still shows the unique consistency as far as the number of peaks is concerned, although here the computational time taken is more than that of others.

7.3.2 Ten-Variable Hump Problem

Next, we consider 10-variable version of the hump function. Here, all parameters remain the same, except the niching parameter. For this problem, the population size for 20, 30, 40 and 50-peak problems are chosen to be 1,200, 1,300, 1,400 and 1,500, respectively. In the deterministic crowding approach, the population size is fixed to be 400 and the number of generations taken are 300, 325, 350, and 375, respectively. The performance of all algorithms is shown in Table 4. The table shows that restricted tournament selection, deterministic crowding, clearing and modified clearing methods are better, although the deterministic crowding approach is not able to maintain enough peaks as compared to other three successful methods. It is also clear that in terms of finding the optima reliably, the modified clearing method is ahead of all others, but with a somewhat larger computational time.

Table 4: Ten-variable hump function.

K		A1	A2	A3	A4	A5	A6	A7	A8
20	SR	17.5	2.62	17.8	0.0	19.8	1.30	0.20	**20.0**
	SD	2.92	1.84	2.28	0.0	0.51	1.68	0.00	**0.0**
	TT	22.5	16.6	**11.8**	-.-	28.4	86.4	243.6	1322.9
	SD	1.56	5.17	**1.18**	-.-	2.15	6.63	274.2	2144.3
30	SR	26.6	2.86	17.9	0.0	29.5	1.24	0.2	**30.0**
	SD	0.91	1.92	3.72	0.0	1.13	0.93	0.00	**0.0**
	TT	30.8	29.0	**16.4**	-.-	54.3	101.7	170.7	1526.8
	SD	3.92	16.8	**0.28**	-.-	23.3	6.04	97.6	1991.2
40	SR	37.6	2.86	25.4	0.0	38.2	1.06	0.20	**40.0**
	SD	2.90	2.04	5.29	0.0	1.08	1.52	0.00	**0.0**
	TT	42.3	36.2	**32.6**	-.-	75.9	335.1	136.6	1893.7
	SD	3.02	31.4	**6.35**	-.-	5.97	136.5	54.0	2257.1
50	SR	47.1	3.18	33.1	0.0	47.5	0.0	0.20	**50.0**
	SD	2.24	2.55	6.63	0.0	4.45	0.0	0.00	**0.0**
	TT	54.0	44.9	**33.6**	-.-	83.9	-.-	154.5	2379.6
	SD	11.3	15.4	**3.25**	-.-	22.8	-.-	138.7	2174.0

7.3.3 25-Variable Hump Problem

Finally, we consider the 25-variable hump problem. By no means, this is an easy task. To investigate the real winner, we increase the population size to 3,000 and run all GAs till 100 generations. In this problem, the radius of each peak is chosen to be 1.45 and all peaks are considered to have an equal height of 1.0, and the shape of the landscape is triangular. We have performed simulations with all eight methods and observed that all algorithms except clearing methods are not able to be find even a single optimum, whereas the clearing approach finds about 43 peaks and the modified clearing approach is able to find all 50 optima. Since most algorithms did not find a single optimum, we do not present the results in a tabular format.

8. CONCLUSIONS

In this paper, we have compared seven existing niche-preserving evolutionary algorithms with a proposed modified clearing strategy in finding and maintaining multiple optimal solutions. Three problems are chosen for this purpose, of which one is a scalable test problems providing a simple way to generate test problems having any number of optima and any shape of optimal basin. Two metrics – number of obtained optima and computational time – are used as a performance measure. Results have shown that various niching algorithms improve the algorithms' ability to maintain stable subpopulations at significant peaks. Among the investigated niching methods, most methods are able to do fairly well, and restricted tournament selection approach, deterministic crowding approach, original clearing approach and the proposed modified clearing method have shown to exhibit better niching property compared to other approaches.

As the number of optima is increased and the dimension of the search space is increased the deterministic crowding approach and the restricted tournament selection approach have not performed well. However, both clearing approaches show better performance. For the 25-variable problem having 50 peaks, the modified clearing approach is the only procedure (of the eight algorithms studied here) which has been able to find *all* global optimal solutions in the hump problem on all 50 simulation runs.

The success of the modified clearing approach comes with a computational burden of performing the clearing approach twice in every generation. However, the method has been found to provide consistent performance of finding all optima present in a problem. These results are interesting and should provide enough clues to researchers to develop new and more efficient algorithms for multi-modal optimization.

9. REFERENCES

[1] K. De Jong. An analysis of the behavior of a class of genetic adaptive systems. Technical report, University Microfilms No. 76-9381,Doctoral dissertation, University of Michigan, 1975.

[2] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Chichester, UK: Wiley, 2001.

[3] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

[4] K. Deb and D. Goldberg. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the Third International Conference on Genetic Algorithms(ICGA-89)*, pages 42–50, 1987.

[5] D. Goldberg. Genetic algorithms in search, optimization and machine learning. reading. Technical report, Illinois Genetic Algorithms Laboratory (IlliGAL), 1989.

[6] D. Goldberg and J. Richardson. Genetic algorithms with sharing for multi-modal function optimization. In *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms (ICGA-87)*, pages 44–49, 1987.

[7] D. E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[8] G. Harick. Finding multi-modal solutions in problems of bounded difficulty. Technical report, Illinois Genetic Algorithms Laboratory, report No. 94002, 1994.

[9] G. Harick. Finding multi-modal solutions using restricted tournament selection. In *Proceedings of the Sixth International Conference on Genetic Algorithms(ICGA-95)*, pages 24–31, 1997.

[10] J.-P. Li, M. E.B., G. T.P., and C. P.J. A species conserving genetic algorithm for multi-modal function optimization. In *Evolutionary Computation*, volume 10(3), pages 207–234, 2002.

[11] S. Mahfoud. Crowding and preselection revisited. In *Parallel Problem Solving from Nature 2*, pages 27–37, 1992.

[12] S. Mahfoud. Simple analytical models of genetic algorithms for multi-modal function optimization. Technical report, Department of Computer Science, University of Lllinois at Urbana-Champaign, Urbana, IL, USA, Illnois Genetic Algorithm Laboratory Report No. 94005, 1993.

[13] S. Mahfoud. Niching method for genetic algorithms. doctoral dissertation. Technical report, Department of Computer Science, University of Lllinois at Urbana-Champaign, Urbana, IL, USA, Illinois Genetic Algorithms Laboratory (IlliGAL) report No. 95001, 1995.

[14] O. Mengsheol and D. Goldberg. Probabilistic crowding: Deterministic crowding with probabilistic replacement. In *Proceedings of the Genetic and Evolutionary Computation Conference-1999(GECCO-99)*, pages 409–416, 1999.

[15] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of Third IEEE International Conference on Evolutionary Computation(ICEC'96)*, pages 798–803. Piscataway, NJ:IEEE Press, 1996.

[16] A. Petrowski. An efficient hierarchical clustering technique for speciation. evolution. In *Artificielle-1997*, pages 22–24, 1997.

[17] A. Petrowski. An efficient hierarchical clustering technique for speciation. evolution. Technical report, Institute National des Telecommunications, Evry, France, Technique Report, 1997.

[18] B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. *Transactions on Evolutionary Computation*, 2(3):97–106, 1998.

[19] X. Yin and N. Germay. Investigations on solving load flow problems by genetic algorithms. In *Electric Power System Research*, pages 151–163, 1991.

[20] X. Yin and N. Germay. 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*, pages 450–457, 1993.