Effect of a Push Operator in Genetic Algorithms for Multimodal Optimization

Yashesh Dhebar[1(✉)] and Kalyanmoy Deb[2]

[1] Department of Mechanical Engineering, Michigan State University, East Lansing, USA
dhebarya@egr.msu.edu

[2] Department of Electrical and Computer Engineering, BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, USA
kdeb@egr.msu.edu
http://www.coin-laboratory.com, https://www.beacon-center.org

Abstract. Genetic Algorithms have been successfully used to solve multi-modal optimization problems, mainly due to their population approach and implicit parallel processing among multiple subpopulations. In order to find and maintain multiple regions, GAs implement a niching principle motivated from nature. The selection procedure of a GA is modified by restricting a comparison among similar solutions to bring about an additional level of diversity in the population. In another recent study, a real-parameter push-operator based on non-uniform coding principle applied to binary-coded GAs was proposed. The push-operator has shown to exhibit better convergence properties on many optimization problems compared to standard GA implementations. In this paper, we extend the push-operator and its implementation with the niching principle to solve multi-modal problems. On a number of constrained and unconstrained multi-modal test problems, we demonstrate its superior convergence to multiple optimal solutions simultaneously. Results are interesting and motivate us to extend the push-operator to multi-objective and other complex optimization tasks.

Keywords: Multimodal optimization · Genetic algorithm · Non-uniform push operator · Niching

1 Introduction

A multimodal optimization task demands a suitable algorithm to find not one, but as many different optimal solutions (local and/or global) as possible. Since evolutionary algorithms (EAs) use a population of points in every iteration, they have always been considered a viable and unique candidate for the task. Since Goldberg and Richardson's 1987 sharing function study, many new methods and applications have been pursued [8,10,11,13–16]. In order to maintain multiple optimal solutions in a population, most studies have modified the selection operator and attempted to create a niche for every disparate optimum discovered in

J.K. Mandal et al. (Eds.): CICBA 2017, Part I, CCIS 775, pp. 3–21, 2017.
DOI: 10.1007/978-981-10-6427-2_1

the EA population. The rise of evolutionary multi-objective optimization [2,4] lies on the success of evolutionary multimodal optimization algorithms, which preceded the former studies.

A recent study on non-uniform mapping of binary-coded genetic algorithms (GAs) helped create a new *push* operator, that can be applied as an additional operator to the offspring solutions obtained after recombination and mutation operators [6]. For optimization tasks of finding a single optimal solution, the push operator simply pushes an offspring solution towards the best-so-far solution in order to enhance the convergence behavior of the overall algorithm. Based on the success of this existing study and a recent preliminary study [5], in this paper, we propose a push operator based EA for finding and maintaining multiple optimal solutions in multimodal problems. Unlike in the existing study, here there are multiple best-so-far solutions in a population and the push operator needs to be applied towards an appropriate best-so-far solution (called a 'leader'). The extension of the original idea for multimodal problems is not straightforward, but we propose an implementation which seems to work well on many constrained and unconstrained multimodal problems.

In the remainder of the paper, Sect. 2 presents the concept of non-uniform coding in binary-coded GAs and the origin of the push operator. Section 3 describes the proposed population-based evolutionary multimodal optimization algorithm in details. Thereafter, Sect. 4 presents the results obtained unconstrained multimodal test problems. Results on constrained multimodal problems are discussed in Sect. 5. An adaptive methodology is then discussed and results on a few problems are presented in Sect. 6. Finally, conclusions are drawn in Sect. 7.

2 Non-uniform Coding and Push Operator

The work done in this research is influenced by the work carried out on non-uniform mapping in binary-coded GAs [6,17]. Traditionally, binary-coded GAs use a uniform mapping in which the gap between decoded values of any two consecutive binary strings is equal. The origin of non-uniform mapping came from the knowledge that the optimum is likely to lie in a certain region of the search region, but it is not guaranteed. It was then argued that a non-uniform mapping with a larger concentration of strings at the likely region should perform better than a uniform mapping representation scheme. While this was demonstrated for binary-coded GAs, an extension of the non-uniform mapping was also envisaged for real-coded GAs [17]. Since there is no string or discrete mapping in real-coded representation, the idea of biased concentration of solutions was introduced through a push operator, which moves the current real-valued variable (x) towards the current-best solution (x_b) in the hope of improving the original variable vector (x), as follows:

$$x' = \begin{cases} a + [(x-a)(x_b-a)^{\eta}]^{\frac{1}{1+\eta}}, & \text{if } x \le x_b, \\ b - [(b-x)(b-x_b)^{\eta}]^{\frac{1}{1+\eta}}, & \text{if } x > x_b. \end{cases} \tag{1}$$

where a is the lower-bound, b is the upper-bound, and η is a parameter chosen to control the amount of push. The idea of this concept is depicted in Fig. 1.

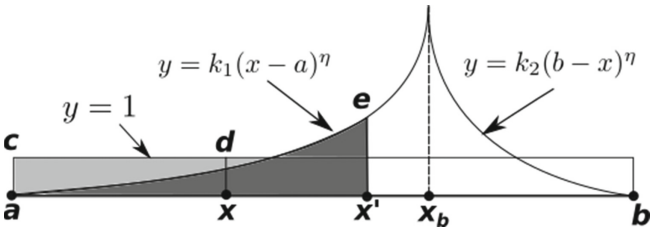

Fig. 1. Non-uniform mapping procedure.

By using a non-uniform polynomial probability function involving order η and a constant (k_1 or k_2 on left or right side), the pushed point x' is calculated by equating the cumulative probability of original point x being chosen using a uniform probability to the cumulative probability of the new point x' being computed using the polynomial probability function. The pushing ensures that the extreme solutions a and b, and best-so-far solution x_b stay at their own locations after the push, whereas all other points move towards the x_b point after the push.

The above push operator was applied to each offspring as an additional GA operator after the recombination and mutation operators are applied. The past study [17] showed that the push operator produced a significant improvement in convergence for a proper choice of the η parameter. Figure 2 shows results on two functions – one unimodal and one multimodal.

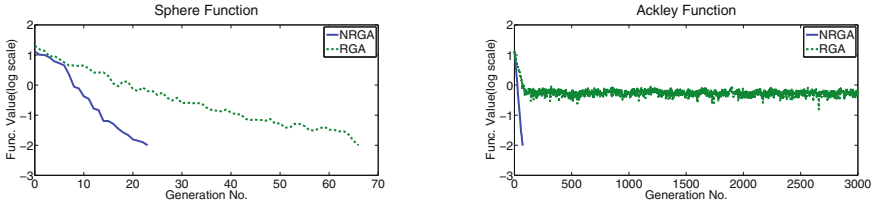

Fig. 2. Improved convergence for a real-parameter GA on $20-D$ Sphere and Ackley functions [17]. NRGA is the non-uniformly mapped RGA.

It is worth noting here that the above push operator may appear to be similar to the operators in particle swarm optimization (PSO) [1] algorithm. Although the pushing towards the current best population member has a similar purpose, the extent of push in our approach is nonlinear and dependent on the location of a population member from the current best solution. Moreover, the push operator here is motivated from a non-uniform mapping in binary-coded GAs.

3 Evolutionary Multimodal Optimization with Push Operator

The aim in a multimodal optimization task is to capture more than one optimum points from the search space in a single simulation run. In this study, we restrict ourselves in finding multiple global optimal solutions (having almost identical objective function values), instead of finding multiple local and global optimal solutions. In the optimization literature, several approaches have been proposed to achieve this which include sharing function method [9], clearing method [14], clustering method [16], and others. These methods modify the selection operator of an EA by discouraging distant solutions (in the variable space) to be compared in the selection process. Since, like solutions are allowed to be compared, multiple niches can form in the population, thereby finding and maintaining multiple optimal solutions through generations. These so-called *niching* methods require a niching parameter that defines a threshold of similarity beyond which the solutions are said to be distant from each other. Some methods to calibrate the niching parameter also exist [7]. In [4], a recommended value for σ_{share} to capture q optimal points in a d dimensional search space is given by the following formula

$$\sigma_{share} = \frac{\sqrt{\Sigma_{k=1}^{d}(x_k^{(U)} - x_k^{(L)})^2}}{2\sqrt[d]{q}}. \tag{2}$$

Here, we intuitively assume that all optimal points are scattered uniformly over the d-dimensional hypercube and are σ_{share} distance away from each other. The hypercube is bounded by the bounds of the variables ($x_k^{(U)}$ and $x_k^{(L)}$). Hence, one can efficiently do a uni-modal optimization by setting σ_{share} to a very large value. If we normalize each dimension by dividing it with the difference of corresponding upper and lower bounds, then we can set the σ_{share} value as

$$\sigma_{share} = \frac{0.5}{\sqrt[d]{q}}. \tag{3}$$

Besides these early studies, a number of more sophisticated studies exist including a multi-objective approach for multimodal optimization [8]. A recent survey on evolutionary multimodal algorithms is a comprehensive source on the topic.

In this section, motivated by the success of the push-operator for RGAs, we present a push-operator based EA for multimodal optimization. It was discussed in the previous section that the push operator requires the best-so-far solution for other population members to be pushed towards it. However, in multimodal problems, there are multiple best solutions, each corresponding to a representative solution of a different optimal basin. Thus, before the push operator can be applied to any solution, a respective best-so-far solution first needs to be identified. For this purpose, we propose a systematic methodology, which is illustrated in Fig. 3.

At every generation t, the population is analyzed to find not one, but multiple, *leaders* (best-so-far solutions) having at least a specific dissimilarity in the

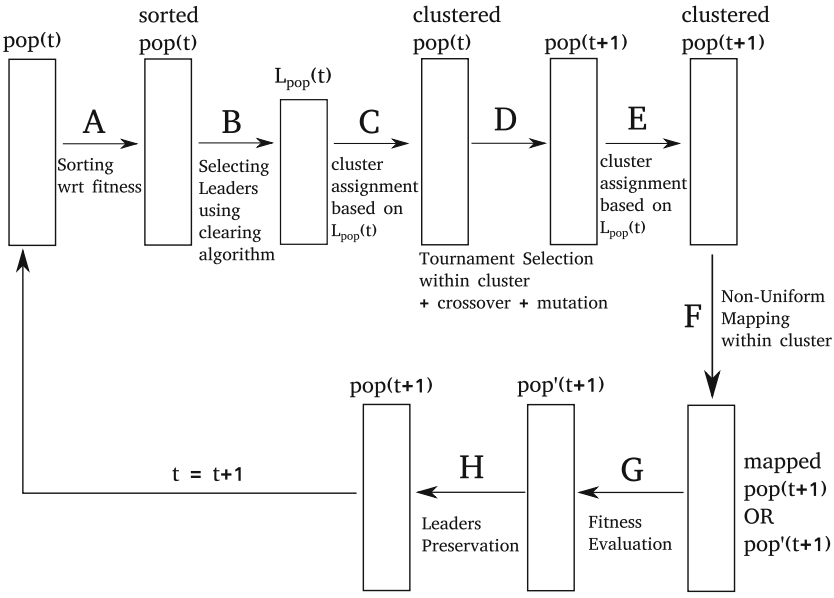

Fig. 3. Flowchart of the proposed multimodal optimization algorithm.

variable space. Every non-leader population member is then associated with a leader to form individual clusters around each leader. Thereafter, a tournament selection operation is executed cluster-wise so that a niche can be formed around each leader. The mating pool is then sent for usual recombination and mutation operations and an offspring population of the size same as the parent population size is created. Before computing the fitness of offspring members, two operations are performed on them. First, offspring members are associated with an existing leader based on their variable-space similarity with the leaders. Once an offspring is associated with a specific leader, the push operator can be applied to move the offspring towards the leader in the hope of improving it. Now, the pushed offspring solutions are evaluated to compute their fitness values. The existing leader population is then combined with the offspring population and the best set of N members (where N is the population size) is chosen as the new population. This ensures preservation and maintenance of leaders from one generation to another. The process is repeated for the new generation. We describe each of the key operations in the next few subsections.

3.1 Selection of Leaders

First, at generation t, selection of leaders is achieved using the clearing algorithm [14]. First, the population is sorted according to its fitness score in acceding order (for a minimization problem) to generate the *sorted population* (S_{pop}). The first member of this sorted-population is automatically considered as the first leader

and is copied to the *leaders-archive* (L_{pop}). Thus, we have $S_{pop}[0] \rightarrow L_{pop}[0]$. We then systematically conduct a search for a member in S_{pop} which is σ_{share} away from the first leader and append it to L_{pop}. The process in repeated to form new leaders and populate L_{pop} with a rule that no two members in L_{pop} are within σ_{share} and $L_{pop}[i]$ dominates $L_{pop}[j]$ in terms of fitness value if $i < j$. The process is terminated if there is no member left in S_{pop} which is σ_{share} away from all individuals present in L_{pop} or the size of L_{pop} reaches its upper-threshold, which in our case is set to $2\times$ *(desired number of optimal points)*. This process is systematically represented in Algorithm 1. This procedure is marked as A and B in Fig. 3.

Sort population P of size n to generate S_{pop}
$L[0] \leftarrow S_{pop}[0]$
$l_{fill} = 1, \qquad i = 1$
for $i < n$ **do**
  if $dist(S_{pop}[i], L[j]) \geq \sigma_{share}, \qquad \forall j \in [0, 1, 2, \ldots, l_{fill} - 1]$ **then**
    $L[l_{fill}] \leftarrow S_{pop}[i]$;
    $l_{fill} = l_{fill} + 1$;
    if $l_{fill} == max_{leaders}$ **then**
      break;
    end if
  end if
  i++
end for

Algorithm 1. Selection of leaders using the clearing algorithm.

3.2 Cluster Assignment

Once we have identified all leaders, we proceed to assign a *cluster-index* (or *c-index*) to the members of the population. Each leader in L_{pop} has a unique *c-index* associated with it. Cluster assignment is about assigning population members their *c-index* based on certain rule. Members with same *c-index* belongs to the same cluster family. In this paper, we carry out the cluster-assignment based on *nearest leader within* σ_{share}. In this technique, an individual is assigned a *c-index* if and only if the nearest leader to it is within σ_{share} distance. With this exercise, there is a possibility to have some individuals not getting associated with any existing leader. Such individuals lie at least σ_{share} away from all leaders present in L_{pop}. These individuals collectively form a new cluster ($U_{cluster}$) which is devoid of any leader. This procedure is marked as C in Fig. 3.

3.3 Localized Selection

Selection operation for each cluster is carried out separately. Each cluster (apart from $U_{cluster}$) has a leader associated to it. Leaders of different clusters are

separated from each other by at least σ_{share} distance. Within a cluster, *tournament selection* is applied to select individuals. Once the above selection operator is executed on all clusters (including $U_{cluster}$), recombination and mutation are carried out as usual over the entire population. This combined process is marked as D in Fig. 3.

The above cluster-wise selection enables us to maintain segregation of population members by avoiding merging of clusters at the time of selection stage. Population generated after mutation undergoes another clustering (or cluster assignment) using the leaders from existing L_{pop} as shown as operation E in Fig. 3.

3.4 Push Operator

At this stage, a non-uniform push operation is carried out for each cluster member keeping the leader of that cluster as the respective best-so-far point. Devoid of any leader, members of $U_{cluster}$ do not undergo any non-uniform push operation. It is important to note that as a result of non-uniform push operation, the density of solutions around each leader increases. For multimodal optimization problems, it is necessary to preserve diversity, especially during initial generations. Thus, having certain portion of population (which for our case is $U_{cluster}$) deprived of participating in the non-uniform push operation is advantageous. Interestingly, from experiments, it is observed that size of $U_{cluster}$ reduces with generations and under some parameter settings, it reaches to zero, thereby indicating that although initially many un-associated population members are around and provide diversity of the population, in later generations most population members have a destiny and are associated with a specific leader. This combined process is marked as F in Fig. 3.

3.5 Preservation of Leaders

The operation G in Fig. 3 evaluates each offspring member. Thereafter, operation H forms the new population by preserving the existing leaders and by choosing any good solutions present in the offspring population. Number of leaders in L_{pop} provides us with a qualitative estimation of segregation in the population. For multimodal optimization problems, it is desired have some diversity preservation property in the optimization algorithm so that the population can reach multiple separated optima. One way to achieve this is by preserving leaders. The approach is pretty straightforward. First, the pushed offspring population and the existing leaders archive (L_{pop}) are merged to form M_{pop}. Then, M_{pop} is sorted according to fitness values and the best N (where N is the population size) members are chosen to form the next generation population.

4 Results on Unconstrained Multimodal Problems

To validate the performance of our algorithm, we test it on eight standard test problems listed below:

F_1 Uniformly Distributed Equal Maxima (1-D)
F_2 Non-Uniformly Distributed Equal Maxima (1-D)
F_3 Himmelblau Function (2-D)
F_4 Six-Hump Camel Back (2-D)
F_5 Modified Rastrigin - 12 Peaks (2-D)
F_6 Modified Rastrigin - 48 Peaks (16-D)
F_7 CMMP(2, 4, 0) - 2-D constrained
F_8 CMMP(10, 16, 0) - 10-D constrained

Problems F_1 to F_6 are solved for maximization and problems F_7–F_8 are solved for minimization of the respective objective function. Problems F_6, F_7 and F_8 are adapted from [8] and problems F_1, F_3, F_4 and F_5 are taken from [12]. Parameter settings for a D-dimensional problem with q desired global optima are mentioned below:

- Population size:
  - 50 for 1-D problems
  - 100 for 2-D problems
  - 480 for F_6 (i.e. $10 \times q$)
  - 250 for CMMP(10, 16, 0)
- Mutation Probability:
  - 0.1 for $F_1 - F_5$ and CMMP(2, 4, 0), CMMP(10, 16, 0)
  - 0.0625 for F_6 (i.e. $1/D$)
- Recombination Probability: 0.9
- SBX Parameters
  - $(\eta_c, \eta_m) = (20, 15)$ for F_1 to F_7
  - $(\eta_c, \eta_m) = (100, 100)$ for CMMP(10, 16, 0)

Selection is carried out with a *binary tournament selection* operator. 50 runs are performed for all problems except for F_6 and F_8 (for which 25 runs are executed). A run is considered *successful*, if all known global optima are found within an accuracy level of 0.01 in the

- objective space for problems from F_1 to F_5 and
- search- space for problems F_6 to F_8.

For all successful runs, the minimum, maximum, median, and average number of function evaluations required to find all optima are reported. On the other hand, a run is labeled as *failure*, when the algorithm fails to reach all peaks even after completing maximum allotted number of function evaluations. For these runs, statistics of the number of global optima reached is reported. Allowable number of function evaluations (FE) for different problems is tabulated in Table 1.

Another metric which is used to measure the performance of the algorithm is the *peak-ratio* (PR), which is calculated as per the directives given in [12]. The governing equation to compute it is mentioned below

$$PR = \frac{\sum_{i=1}^{NR} NPF_i}{NKP * NR}, \tag{4}$$

Table 1. Maximum number of function evaluations allowed.

Function	Max. FEs	Function	Max. FEs
$F_1 - F_4$	$5.0E{+}04$	F_7	$5.0E{+}04$
F_5	$2.0E{+}05$	F_8	$6.25E{+}05$
F_6	$4.8E{+}05$		

where NR is the total number of runs, NPF_i is total number of global optima reached in i-th run and NKP is the actual number of global optima. As mentioned before, all results are computed using the push-enabled multimodal RGA. η is updated linearly with the rate of $\bar{\eta}/max_{generations}$ per generation. For comparison with RGA, we run the same algorithm with $\bar{\eta}$ set to 0, thereby creating an algorithm devoid of any push operation.

4.1 F_1: Five Equal Maxima Function

$$F_1(x) = \sin^6(5\pi x). \tag{5}$$

This problem has only one variable, bounded between 0 and 1, and has five maxima. Results for this problem are shown in Table 2. It can be seen that in all cases, $PR = 1$ is obtained, meaning that all known peaks are successfully found. On increasing the value of $\bar{\eta}$, the performance in terms of average number of function evaluations gets better. The best performance is observed when the rate of increase in η is 0.02/generation (or equivalently $\bar{\eta} = 20$ for 1,000 generations). But, if the rate is increased to a higher value than 0.02, the performance deteriorates. This is due to the fact that by increasing the rate, more push is made towards the current leaders, which leads to the depletion of diversity during initial generations. Due to lack of diversity, exploration of the search space becomes a difficult task and hence it requires relatively more number of generations to converge.

Table 2. Results for function F_1 (1-D with five uniformly distributed global maxima).

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.000/(0.0)	S = 50	101	251	378.00	5751	1.00
0.005/(5.0)	S = 50	101	251	388.00	5801	1.00
0.010/(10.0)	S = 50	101	251	292.00	1801	1.00
0.020/(20.0)	*S = 50*	*101*	*251*	*273.00*	*651*	*1.00*
0.030/(30.0)	S = 50	101	251	279.00	701	1.00
0.040/(40.0)	S = 50	101	251	1515.00	48451	1.00
0.050/(50.0)	S = 50	101	251	1129.00	36301	1.00

4.2 F_2: Non-uniformly Distributed Maxima Function

The next function we consider is derived from the previous function. We tweak the function in Eq. 5 by making the argument of the sinusoidal function as a quadratic in x, thereby resulting into the following functional form

$$F_2(x) = \sin^6(3\pi x^2). \tag{6}$$

Distribution of three optimal points for this function is shown in Fig. 4. Results in Table 3 show that non-uniform push makes the convergence faster, with best performing simulation observed when η is increased at a rate of 0.02/generation.

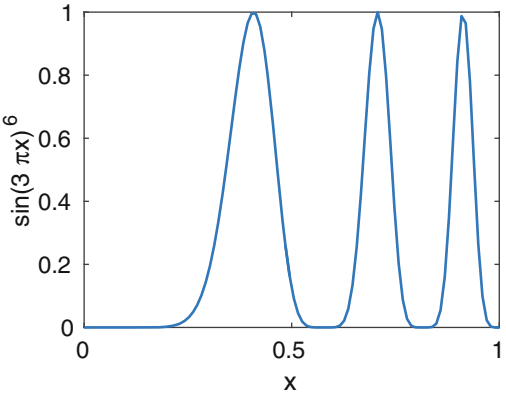

Fig. 4. Non-uniformly distributed optimal points with three global maxima.

Table 3. Results for function F_2 (1D with 3 non-uniformly distributed global maxima)

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	51	201	351.00	7051	1.00
0.01/(10.0)	S = 50	51	201	205.00	401	1.00
0.02/(20.0)	*S = 50*	*51*	*201*	*202.00*	*351*	*1.00*
0.03/(30.0)	S = 50	51	201	206.00	451	1.00
0.04/(40.0)	S = 50	51	201	215.00	451	1.00
0.05/(50.0)	S = 50	51	201	210.00	451	1.00

4.3 F_3: Himmelblau Function

This problem has four global maxima distributed over a 2-D domain

$$F_3(\mathbf{x}) = 200 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2. \tag{7}$$

Both variables are bounded between −6 and 6. Results for this problem are tabulated in Table 4. The algorithm shows consistent improvement in the speed of convergence as $\bar{\eta}$ is increased from 0 to 50. To further investigate the effect of the push-operator, three extra set of experiments are carried out with $\bar{\eta} = 100, 200$ and 1000. The optimal performance can now be seen when the rate of change in η is 0.4/gen (or $\bar{\eta} = 200$). For this, 65.85% reduction in the median number of function evaluations (from 3,701 to 1,301) is observed. But, for too large a value of $\bar{\eta} = 1,000$, the performance deteriorates significantly.

Table 4. Results on Himmelblau function F_3 (2-D, four global maxima).

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	1701	3701	3839.00	8101	1.00
0.02/(10.0)	S = 50	1701	2501	2581.00	6901	1.00
0.04/(20.0)	S = 50	1401	2101	2239.00	6001	1.00
0.06/(30.0)	S = 50	1201	1801	1917.00	3201	1.00
0.08/(40.0)	S = 50	801	1601	1691.00	3901	1.00
0.10/(50.0)	*S = 50*	*1001*	*1501*	*1755.00*	*4001*	*1.00*
0.20/(100.0)	S = 49	801	1301	1496.92	4301	0.99
	F = 1	3	3	3.00	3	
0.40/(200.0)	**S = 50**	**801**	**1301**	**1441.00**	**3601**	**1.00**
2.00/(1000.0)	S = 26	601	1301	1470.23	3101	0.87
	F = 24	2	3	2.92	3	

4.4 F_4: Six Hump Camel Back Function

This problem has two variables. Its expression is given below

$$F_4(\mathbf{x}) = -4\left[\left(4 - 2.1x^2 + \frac{x^4}{3}\right)x^2 + xy + (4y^2 - 4)y^2\right], \tag{8}$$

where $x \in [-1.9, 1.9]$ and $y \in [-1.1, -1.1]$. The problem has two global and two local maxima. From Table 5, we can observe that the non-uniform push aids to achieve faster convergence and the performance improves upon increasing the value of $\bar{\eta}$ from 0 to 50. This trend is also followed even for higher values of $\bar{\eta} = 100$, 200, or 1,000. For the best performing simulation, the median number of function evaluations required to find all optimal points is reduced by about 74.94% (301 as compared to 1,201).

4.5 F_5: 2-D Modified Rastrigin Function

The generalized formula for a modified Rastrigin function is given below:

$$F_5(\mathbf{x}) = -\sum_{i=1}^{D}(10 + 9\cos(2\pi k_i x_i)). \tag{9}$$

Table 5. Results for function F_4, Six Hump Camel Back (2-D with two global and 2 two local maxima).

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	501	1201	1223.00	3701	1.00
0.02/(10.0)	S = 50	401	801	941.00	2001	1.00
0.04/(20.0)	S = 50	401	901	873.00	1601	1.00
0.06/(30.0)	S = 50	501	801	767.00	1201	1.00
0.08/(40.0)	S = 50	401	701	709.00	1001	1.00
0.10/(50.0)	*S = 50*	*501*	*701*	*689.00*	*1001*	*1.00*
0.20/(100.0)	S = 50	301	601	561.00	1001	1.00
0.40/(200.0)	S = 50	301	501	481.00	701	1.00
2.00/(1000.0)	**S = 50**	**201**	**301**	**395.00**	**1001**	**1.00**

For $2D$ case, we fix $k_1 = 3$ and $k_2 = 4$, which makes a landscape comprising of 12 peaks, when x_1 *and* $x_2 \in [0, 1]$. Results for this function are tabulated in Table 6. Similar to the behavior we observed in F_3 and F_4, here too, we observe the monotonic reduction in the number of function evaluations upon increasing the rate of change of η from 0.00 to 0.04. However, this trend is no longer followed when we set $\bar{\eta}$ to higher values and the plateau in performance is hit for $\bar{\eta} = 120$.

Table 6. Results for function F_5, 2-D modified Rastrigin with 12 maxima.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.000/(0.0)	S = 50	1601	2201	2361.00	4501	1.00
0.005/(10.0)	S = 50	1501	2101	2139.00	3401	1.00
0.010/(20.0)	S = 50	1301	2101	2137.00	3901	1.00
0.020/(40.0)	S = 50	1401	1901	1963.00	3701	1.00
0.030/(60.0)	S = 50	1201	1701	1783.00	3201	1.00
0.040/(80.0)	*S = 50*	*1201*	*1701*	*1689.00*	*2501*	*1.00*
0.050/(100.0)	S = 50	1201	1601	1725.00	2801	1.00
0.060/(120.0)	**S = 50**	**1201**	**1601**	**1629.00**	**3001**	**1.00**
0.250/(500.0)	S = 47	1001	1501	1564.83	2401	0.985
	F = 3	9	9	9.00	9	
0.500/(1000.0)	S = 47	801	1901	3111.64	49401	0.983
	F = 3	8	9	8.67	9	

4.6 F_6: 16-D Modified Rastrigin Function

This problem has 16 variables between 0 and 1

$$F_6(\mathbf{x}) = -\sum_{i=1}^{D}(10 + 9\cos(2\pi k_i x_i)). \tag{10}$$

Value of $k_i's$ are chosen as follows:

$$k_4 = k_8 = 2,\ k_{12} = 3,\ k_{16} = 4,\ k_i = 1, \text{ for } i = 1{-}3, 5{-}7, 9{-}11, 13{-}15.$$

For the given domain, we have 48 optimal points over a 16-dimensional space. Initially, with the recommended value of σ_{share} of 0.3925 (calculated using Eq. 3) it was not possible to converge at all peaks. This is because the value of σ_{share} calculated using Eq. 3 assumes uniform distribution of optimal points in the entire search space. In this problem, we have highly uneven distribution of optimal points with minimum separation being only 0.25. Thus, the value of $\sigma_{share} = 0.3925$ was too high and eventually resulted in merging of multiple clusters into one. To counter this, we reduce the value of σ_{share} to 0.125. Results shown in Table 7 indicate that our algorithm is able to capture all 48 optimal points. Best performing runs are conducted with $\bar{\eta} = 30$ (or when the rate of increase in η was 0.03 per generation) where it took 1,920 less median number of function evaluations.

Table 7. Results for function F_6, 16-D modified Rastrigin with 48 global minima.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 25	16321	19681	21697.00	61921	1.00
0.01/(10.0)	S = 25	14881	18721	19623.40	37441	1.00
0.02/(20.0)	S = 25	14881	17761	19853.80	39841	1.00
0.03/(30.0)	*S = 25*	*13921*	*17761*	*19412.20*	*49921*	*1.00*
0.04/(40.0)	S = 25	14881	18241	19354.60	47521	1.00
0.05/(50.0)	S = 25	15841	17281	20007.40	56641	1.00

5 Constrained Multimodal Problems

Apart from the unconstrained problems, we also investigated the behavior of our algorithm for a constrained case. Two test problems were considered: *CMMP(2, 4, 0)* and *CMMP(10, 16, 0)*. These were adapted from [8]. The constrained handling was done using a parameter-less approach [3].

Also, before the leaders preservation stage, fitness of all constraint-violating individuals, including the ones present in the leaders-archive is modified. First, we compute the fitness value of the worst feasible individual in the combined population of current population members and the leaders-archive ($f_{WorstFeasible}$).

Then, the fitness value of infeasible solutions is modified by adding the constraint violation (CV) according to the following equation[1]

$$f_{modified}(\mathbf{x}) = f_{WorstFeasible} + \text{CV}(\mathbf{x}). \tag{11}$$

Results for CMMP(2, 4, 0) and CMMP(10, 16, 0) are shown in Tables 8 and 9, respectively.

Table 8. Results on 2-D, 4-peak CMMP(2, 4, 0) problem.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	701	1401	1537.00	3201	1.00
0.02/(10.0)	S = 50	701	1301	1503.00	3901	1.00
0.04/(20.0)	S = 50	501	1301	1561.00	4401	1.00
0.06/(30.0)	S = 50	601	1101	1487.00	16901	1.00
0.08/(40.0)	*S = 50*	*501*	*1101*	*1379.00*	*4401*	*1.00*
0.10/(50.0)	S = 50	601	1101	2105.00	19501	1.00

Similar to the situation we encountered for problem F_6, for CMMP(10, 16, 0) too, we reduce σ_{share} from its recommended value to 0.04. In all cases, a large proportion of known global optima are found by our proposed algorithm.

Table 9. Results on 10-D, 16-peak CMMP(10, 16, 0) problem.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.000/(0.0)	*S = 25*	*54751*	*74251*	*110441.00*	*469251*	*1.00*
0.002/(5.0)	S = 24	35751	105501	140188.50	593501	0.98
	F = 1	8	8	8.00	8	
0.004/(10.0)	S = 18	34251	86251	131570.44	398501	0.82
	F = 7	0	6	5.57	12	

6 Adaptive Normalization of Niche Radius

In previous section, we encountered a situation where we modified σ_{share} by reducing its value from the recommended one. To avoid this resetting of σ_{share} value, we worked upon adaptively updating the way *normalized distance* is calculated between two points. In all the previous experiments, normalized distance between two points **A** and **B** was computed using the following formula

$$dist(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{d} \left(\frac{A_i - B_i}{U_i - L_i} \right)^2}, \tag{12}$$

[1] For a maximization problem, CV(**x**) will be subtracted.

where A_i's and B_i's are i-th component of $\mathbf{A}$ and $\mathbf{B}$ and U_i's and L_i's are its corresponding upper and lower bounds. Value of σ_{share} was calculated assuming that all optimal points were uniformly scattered in the hypervolume bounded by U_i's and L_i's. Thus, for normalization, we divided the difference of each component with $(U_i - L_i)$. Another assumption can be drawn by intuitively visualizing that optimal points are uniformly located in the region enclosed by the population cloud. To compute the stretch of this region along each dimension, we first generate a set of well-separated individuals (WSI) in the current population. Standard deviation of this set is then calculated and used as a normalizing factor instead of $(U_i - L_i)$. Generation of WSI is similar to that of L_{pop}, but unlike in L_{pop} we do not keep an upper threshold on the number of members set of WSI can have. Once we have set WSI, the procedure to compute the normalized distance is mentioned below

Step 1: Standard deviation of individuals in set WSI across i-th dimension (σ_i) is first calculated.
Step 2: Maximum of these standard deviations is identified as σ_{max}.
Step 3: Normalization factor C_i for each dimension is set as $\min((2\sigma_{max}, (U_i - L_i)))$.
Step 4: Normalized distance between two points $\mathbf{A}$ and $\mathbf{B}$ is calculated using

$$dist(\mathbf{A}, \mathbf{B}) = \sqrt{\sum_{i=1}^{d} \left(\frac{A_i - B_i}{C_i} \right)^2}. \tag{13}$$

It is to note here that we are not explicitly changing the bounds (U_i and L_i), we are just modifying the normalizing factor from $(U_i - L_i)$ to C_i for each dimension.

6.1 Results Using Adaptive Normalization

Results generated using the above adaptive normalization approach are given in Tables 10, 11, 12, 13 and 14 for some of the problems. For comparison, the last row of each table gives the result of the best performing $\bar{\eta}$ for non-adaptive normalization scenario.

One characteristic which clearly became evident from all these studies is that after implementing the adaptive normalization, the algorithm quickly forms as many clusters as there are global optima in the problem. Figure 5 shows this characteristic for a 16-D modified Rastrigin function. This acts as a significant aid for our algorithm by reducing the noise that would have otherwise affected an optimization run.

Table 10. Results with adaptive normalization of niche radius for F_1.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.000/(0.0)	S = 50	151	251	293.00	651	1.00
0.005/(5.0)	S = 50	151	251	302.00	1251	1.00
0.010/(10.0)	S = 50	151	251	300.00	1651	1.00
0.020/(20.0)	S = 49	151	251	290.80	1201	0.99
	F = 1	4	4	4.00	4	
0.030/(30.0)	S = 50	151	301	286.00	551	1.00
0.040/(40.0)	S = 50	151	251	279.00	651	1.00
0.050/(50.0)	*S = 50*	*151*	*251*	*272.00*	*501*	*1.00*
(NA)0.02/(20.0)	*S = 50*	*101*	*251*	*273.00*	*651*	*1.00*

Table 11. Results with adaptive normalization of niche radius for F_5.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	1501	2201	2369.00	3901	1.00
0.01/(20.0)	S = 50	1501	2201	2193.00	3701	1.00
0.02/(40.0)	S = 50	1301	1901	1987.00	4801	1.00
0.03/(60.0)	S = 50	1201	1701	1803.00	2701	1.00
0.04/(80.0)	S = 50	1201	1601	1681.00	2501	1.00
0.05/(100.0)	*S = 50*	*1201*	*1601*	*1671.00*	*2701*	*1.00*
(NA)0.04/(80.0)	*S = 50*	*1201*	*1701*	*1689.00*	*2501*	*1.00*

Table 12. Results with adaptive normalization of niche radius for F_6.

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 25	14881	16321	17300.20	33121	1.00
0.01/(10.0)	S = 25	12001	13921	13767.40	16321	1.00
0.02/(20.0)	S = 25	11041	13921	13805.80	18721	1.00
0.03/(30.0)	*S = 25*	*11041*	*13441*	*13748.20*	*18241*	*1.00*
0.04/(40.0)	S = 25	10561	13921	13959.40	17281	1.00
0.05/(50.0)	S = 25	11521	13921	14074.60	17761	1.00
(NA)0.03/(30.0)	*S = 25*	*13921*	*17761*	*19412.20*	*49921*	*1.00*

Table 13. Results with adaptive normalization of niche radius for CMMP(2, 4, 0).

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.00/(0.0)	S = 50	901	1901	2121.00	5401	1.00
0.02/(10.0)	S = 50	901	1601	1927.00	4901	1.00
0.04/(20.0)	S = 50	801	1801	1855.00	4001	1.00
0.06/(30.0)	S = 50	601	1801	2017.00	4701	1.00
0.08/(40.0)	*S = 50*	*801*	*1601*	*1887.00*	*5801*	*1.00*
0.10/(50.0)	S = 50	801	1701	2103.00	6601	1.00
(NA)0.08/(40.0)	*S = 50*	*501*	*1101*	*1379.00*	*4401*	*1.00*

Table 14. Results with adaptive normalization of niche radius for CMMP(10, 16, 0).

rate/$\bar{\eta}$	S/F	min	med	avg	max	PR
0.000/(0.0)	S = 23	46501	117751	143381.43	462001	0.960
	F = 2	8	8	8.00	8	
0.004/(10.0)	*S = 23*	*24751*	*62001*	*104522.74*	*283501*	*0.960*
	F = 2	*8*	*8*	*8.00*	*8*	
0.008/(20.0)	S = 17	25251	149501	185956.88	452751	0.845
	F = 8	4	8	8.25	14	
0.00/(0.0)	*S = 25*	*54751*	*74251*	*110441.00*	*469251*	*1.00*

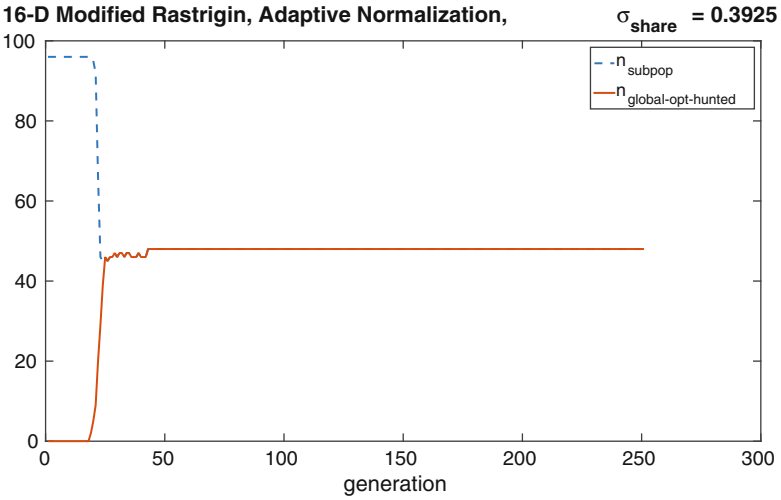

Fig. 5. Number of clusters ($n_{sub-pop}$) formed (dashed line) and number of global optimal points captured ($n_{global-optima-hunted}$) (solid line).

7 Conclusions

This paper has extended a push operator developed from the working principle of a non-uniform binary-coded GA to a real-coded GA for finding a single optimum solution. In solving multimodal problems, the push operator needs to be applied to every population member differently, as there are more than one representative solutions in a population, one for each optimum and every solution must be pushed towards an appropriate one. We have proposed an algorithm which is elite-preserving and it employs a niching operator based on the clearing approach. The push operator aids in moving population members closer to the nearest representative optimal solution (called a 'leader') in a parallel manner towards all discovered leaders. The overall algorithm has been shown to find near 100% global optima for a set of constrained and unconstrained test problems.

We have also extended the algorithm to compute the closeness of two solutions by using an adaptive distance metric, which seems to outperform the original method. The algorithm and results proposed in this study are interesting and motivating to us for applying it to more complex and practical multimodal optimization problems.

Acknowledgments. This material is based in part upon work supported by the National Science Foundation under Cooperative Agreement No. DBI-0939454. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

1. Barrera, J., Coello, C.A.C.: A review of particle swarm optimization methods used for multimodal optimization. In: Lim, C.P., Jain, L.C., Dehuri, S. (eds.) Innovations in Swarm Intelligence, vol. 248, pp. 9–37. Springer, Berlin (2009)
2. Coello, C.A.C., VanVeldhuizen, D.A., Lamont, G.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Boston (2002)
3. Deb, K.: An efficient constraint handling method for genetic algorithms. Comput. Meth. Appl. Mech. Eng. **186**(2), 311–338 (2000)
4. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms, vol. 16. Wiley, Hoboken (2001)
5. Dhebar, Y., Deb, K.: A computationally fast multimodal optimization with push enabled genetic algorithm. In: Proceedings of Genetic and Evolutionary Computation Conference Companion, pp. 191–192. ACM, Chicago (2017)
6. Deb, K., Dhebar, Y.D., Pavan, N.: Non-uniform mapping in binary-coded genetic algorithms. In: Bansal, J., Singh, P., Deep, K., Pant, M., Nagar, A. (eds.) Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012). Advances in Intelligent Systems and Computing, vol. 201, pp. 133–144. Springer, Gwalior (2013)
7. Deb, K., Goldberg, D.E.: An investigation of niche and species formation in genetic function optimization. In: Proceedings of the Third International Conference on Genetic Algorithms, pp. 42–50 (1989)

8. Deb, K., Saha, A.: Multimodal optimization using a bi-objective evolutionary algorithm. Evol. Comput. **20**(1), 27–62 (2012)
9. Goldberg, D.E., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the First International Conference on Genetic Algorithms and Their Applications, pp. 41–49 (1987)
10. Lee, C., Cho, D., Jung, H.: Niching genetic algorithm with restricted competition selection for multimodal function optimization. IEEE Trans. Magn. **35**(3), 1722–1725 (1999)
11. Li, X.: Efficient differential evolution using speciation for multimodal function optimization. In: Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO-2005), pp. 873–880 (2005)
12. Li, X., Engelbrecht, A., Epitropakis, M.G.: Benchmark functions for cec2013 special session and competition on niching methods for multimodal function optimization. RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Technical report (2013)
13. Mengsheol, O., Goldberg, D.E.: Probabilistic crowding: deterministic crowding with probabilistic replacement. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO-1999), pp. 409–416 (1999)
14. Petrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: Proceedings of Third IEEE International Conference on Evolutionary Computation (ICEC 1996), pp. 798–803. IEEE Press, Piscataway (1996)
15. Rönkkönen, J., Lampinen, J.: On determining multiple global optima by differential evolution. In: Proceedings of Evolutionary and Deterministic Methods for Design, Optimization and Control (Eurogen 2007), pp. 146–151 (2007)
16. Streichert, F., Stein, G., Ulmer, H., Zell, A.: A clustering based niching EA for multimodal search spaces. In: Liardet, P., Collet, P., Fonlupt, C., Lutton, E., Schoenauer, M. (eds.) EA 2003. LNCS, vol. 2936, pp. 293–304. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24621-3_24
17. Yashesh, D., Deb, K., Bandaru, S.: Non-uniform mapping in real-coded genetic algorithms. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2237–2244. IEEE (2014)