Crowding-based local Differential Evolution with Speciation-based Memory Archive for Dynamic Multimodal Optimization

Souvik Kundu
Jadavpur University
188 R. S. C. Mullik Road
Kolkata-700 032, India
sk210892@gmail.com

Subhodip Biswas
Jadavpur University
188 R. S. C. Mullik Road
Kolkata-700 032, India
sub17was@gmail.com

Swagatam Das
Indian Statistical Institute
203 B. T. Road
Kolkata-700 108, India
swagatam.das@isical.ac.in

P. N. Suganthan
Nanyang Tech. University
50 Nanyang Avenue
Singapore 639798
epnsugan@ntu.edu.sg

ABSTRACT
In the real world, many problems are multimodal as well as dynamic. This type of problems requires optimizers which not only locate multiple optima in a single run but also track the changing optima positions in the dynamic environments. In this paper a niching parameter free algorithm is designed which can locate multiple optima in changing environments. The proposed algorithm integrates the crowding concept with a competent EA called Differential Evolution for maintaining the multiple peaks in a single run. To avoid the use of niching parameter that requires prior knowledge about the fitness landscape, the authors have used local mutation for searching the solution space. A speciation-based memory archive is integrated for regeneration of population after an environmental change is detected. Experimental analysis is conducted on the Moving Peaks Benchmark problem and the performance of the proposed algorithm is compared with other peer algorithms to highlight the overall effectiveness of our work.

Categories and Subject Descriptors
G.1 [**Numerical Analysis**]: Optimization; F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems.

General Terms: Algorithms, Performance, Experimentation.

Keywords: Dynamic environment, Multimodal optimization, Differential Evolution, niching method, crowding, speciation-based memory.

1. INTRODUCTION
Differential Evolution (DE), first proposed by Storn and Price [1], is a very simple and powerful optimizer. In recent years DE has been increasingly used for solving many complex and difficult optimization problems. The main operational steps in DE are quite similar to the traditional EAs. However, unlike EA, DE-variants perturb the current population by means of scaled difference vectors that are formed by random individuals distinctly picked up from the population of current generation. The classical DE was mainly designed for solving static global optimization problems with the aim of finding the most optimal solution.

GECCO'13, July 6–10, 2013, Amsterdam, The Netherlands.

In later years researchers have extended DE for solving both dynamic and multimodal optimization problems. In case of multimodal problems, multiple optima have to be parallely tracked in a single run. So the algorithm must be designed in a way that it should discover multiple peaks and at the same time should maintain those peaks until the end of the run. To tackle this kind of problems DE has been integrated with several niching methods like crowding, speciation, sharing to form Crowding DE (CDE) [2], Species-based DE (SDE) [3] and such likes. The problems where the functional landscapes change over time are known as Dynamic Optimization Problems (DOPs). For solving this kind of problems the algorithms should track the changing optima or detect the newly emerging optimal solution in the new landscape. In case of DOPs the most challenging task is to maintain the diversity throughout the population or else there will be premature convergence of the population members which will severely inhibit the performance of conventional EAs. In addition to this, the optimization algorithm must have fine searching ability to produce high quality solutions by tracking the changing optima. Although several DE-based algorithms for finding global optima in DOPs [4, 5] have been proposed, there have been hardly much contribution in formulating niching based DEs that operate on uncertain, stochastically changing environments.

Additional challenge is faced by an algorithm in locating and tracking multiple optima in a dynamic environment owing to factors like- shifting location of optima, shift severity, loss of population diversity, etc. Even the height of the peaks may vary and it can also happen that the height of global peak decreases where the height of some local peaks increases. So the major requirements of a niching algorithm to perform well enough in dynamic multimodal environment are:

- Searching multiple peaks in parallel
- Tracking the peak once located
- Responding to change in environment
- Maintaining threshold diversity.

For tackling this kind of dynamic multimodal problem we have considered a restrictive form of mutation that uses local knowledge of adjacent members and integrated it with crowding DE. The mutation is performed by randomly picking the individuals from the local neighborhood, i.e. geographical neighborhood formed by means of lesser Euclidean distance. This helps in inducing intrinsic tendency to maintain peaks. It has been observed that using memory archive the performance of an algorithm in successive change instances can be greatly improved [6]. Getting motivation from this, we have used speciation-based memory archive where at every generation some solutions around

both the global and local peaks are conserved for the next instance.

The rest of the paper has been organized as follows. In Section 2 the related works and the motivations for our research is stated. This is followed by Section 3 dedicated to understanding the dynamics of DE, which is the EA used here. With reference to crowding DE in Section 4, our proposed algorithm is described in Section 5. The experimental setup is stated in Section 6 while the results and analysis is conducted in Section 7. Finally the paper is concluded in Section 8 with directions to research that may be undertaken in near future.

2. RELATED WORKS AND MOTIVATION

Significant researches have been done for solving both dynamic and multimodal problems separately. We review some of the existing works that have inspired the authors to work on dynamic multimodal problems.

2.1 Locating multiple Optima in static Environment

Static multimodal problems have been studied extensively. Different niching methods have been integrated with the DE such as CDE [2], SDE [3], ShDE [2] for tracking multiple optima in a single run. But in most of the well known niching techniques use a niching parameter which is dependent upon the landscape quality. So a prior knowledge is required for the landscape which is not always readily available. This is the main drawback of those algorithms. To overcome this problem recently Suganthan *et al*. [7] proposed neighborhood based mutation. This concept refined the stochastic nature of parent vector selection in DEs and when integrated with the basic standalone techniques, it reported a significant improvement in the overall performance of DE on a variety of static multimodal problems. In this case the offspring is generated within the same niche where the parent belongs. So both the offspring and the parent individual are in close vicinity of each other. Hence it enhances the searching of DE by allowing the mutation to produce offspring in the neighborhood of fitness basin and does away with drifts.

2.2 Tracking optima in dynamic Environment

Most of the works in DE-based algorithms for solving DOPs are designed for tracking a single optimum. Several approaches like multipopulation [8, 4], clustering [9], etc have been adopted in basic DE framework in recent years and satisfactory results have been obtained.

An extra challenge of the EAs involved in dynamic optimization problem is the detection of environment change and responding to the change. Literature presents several schemes for tackling change [10]. Memory based approaches have been proposed to use the historic information [11, 12]. It has been observed that using information from previous generations helps to detect optima when it is not far from previous locations. Yang and Yao [13] proposed an incremental learning scheme with associative memory in GAs for solving DOPs.

Now to detect multiple optima in dynamic environment several particle swarm-based algorithms have been proposed. Li *et al*. in [14] proposed an improved particle swarm optimizer (PSO) using the notion of species to determine the neighborhood best values. Yang *et al.* investigated a clustering particle swarm optimizer which used a hierarchical clustering method to locate and track multiple peaks in a dynamic environment [9]. However, recently, EAs have been proposed for environments which are both multimodal as well as dynamic [15, 10, 16].

2.3 Motivations

Due to the simplicity and effectiveness of the DE variants in optimization domain, efforts are being made for extending their application to multimodal domain [7] as well as the domain of DOPs [6]. Mendes and Mohais [4] proposed multipopulation in DE for detecting global optimum. But the implementation of DE in devising a niching parameter free framework for locating multiple peaks in dynamically changing multimodal problems is few.

In this research article we have exploited DE, an EA with high potential for tracking global optimum, and extended its search behavior to track and locate multiple optima in non-stationary environments. Usually memory archives bring about marked enhancement in the performance of EAs in DOPs. Keeping it in mind the aforesaid factors, we have proposed a hybridized niching variant of DE that uses a niching method called crowding, to restrict the tendency to drift towards the global optima, and combine it with a speciation-based memory archive to regenerate the population that holds some high quality solutions from previous landscape and merges them with a random portion of present population sample. In what follows, this hybridized DE variant shall be referred to as ClDES (Crowding-based Local DE with Speciation memory). ClDES simultaneously aims at reducing dependency on niching parameter by localized mutation while tackling dynamic multimodality.

3. DIFFERENTIAL EVOLUTION

Differential Evolution, proposed by Storn and Price [1] in 1995, is a population-based simple, efficient heuristic for solving global optimization problems. The basic DE is composed of four main steps which are initialization of trial population members, mutation, crossover and selection between parent and generated offspring. In the first step the trial population is initialized within the search range of the solution space. Let us consider there are *NP D*-dimensional vectors as the members of the trial population. Then d^{th} dimension of the i^{th} individual can be initialized as:

$$X_{i,d} = lb_d + \left(ub_d - lb_d\right) \times rand\left(0,1\right) \tag{1}$$

where ub_d and lb_d are the upper bound and lower bound of the d^{th} dimension of the objective space.

In the mutation phase, DE creates a donor vector $\vec{V}_i^G$ in the G^{th} generation. Although there are different strategies suggested so far, we have used the basic DE/rand/1 mutation strategy since it maintains perfect tradeoff between exploration and exploitation. It can be represented mathematically as:

$$\vec{V}_i^G = \vec{X}_{r_1}^G + F.\left(\vec{X}_{r_2}^G - \vec{X}_{r_3}^G\right) \tag{2}$$

where r_1, r_2 and r_3 are three mutually exclusive random indices ($r_1 \neq r_2 \neq r_3$) in the current generation G and F is the scale factor in [0, 2] for scaling the difference vector. It is advisable to limit the value of F in the range of [0, 0.95] for successful mutation.

Through crossover operation, the generated mutant vector mixes its components with the parent vector to form the trial vector $\vec{U}_i^G = \left\{u_{i,1}^G, u_{i,2}^G, \ldots u_{i,D}^G\right\}$. This vector is formed via binomial crossover. The scheme may be outlined as:

$$u_{i,j}^{G} = \begin{cases} u_{i,j}^{G} & \text{if } rand_{i,j} \le Cr \text{ or } j = j_{rand} \\ x_{i,j}^{G} & \text{otherwise} \end{cases} \quad (3)$$

where Cr is the crossover rate and j_{rand} is the random index in [1, D]. Finally, the selection process, which is performed by means of fitness criterion, can be represented as:

$$\vec{X}_i^{G+1} = \begin{cases} \vec{U}_i^{G} & \text{if } f\left(\vec{U}_i^{G}\right) \le f\left(\vec{X}_i^{G}\right) \\ \vec{X}_i^{G} & \text{if } f\left(\vec{U}_i^{G}\right) > f\left(\vec{X}_i^{G}\right) \end{cases} \quad (4)$$

where $f(.)$ is the objective function to be minimized. In Algorithm 1, the basic DE process has been elucidated for convenience.

***Algorithm* 1**: Classical Differential Evolution.

1 $\vec{X}$ = initialize the population randomly as (1). $\vec{X}$ is a matrix of order $NP \times D$.
2 ***for*** $g=1:G_{max}$
3 ***for*** $i=1:NP$
4 Randomly select three mutant vectors $\vec{X}_{r_1}$, $\vec{X}_{r_2}$ and $\vec{X}_{r_3}$.
5 $\vec{V}_i$ = generate mutant vector as presented in (2).
6 ***for*** $j = 1: D$
7 $u_{i,j}$ = recombine ($x_{i,j}$, $v_{i,j}$) ; as mentioned in (3).
8 ***endfor***.
9 Selection of $\vec{X}$ for the next generation by (4).
10 ***endfor***.
11 ***endfor***.

4. CROWDING AND CROWDING DIFFERENTIAL EVOLUTION

4.1 Crowding

Crowding is a simple niching technique. Researchers introduced additional modifications to the original crowding method in order to improve its performance.

4.1.1 Original Crowding: The crowding method was proposed by De Jong in 1975 [17]. This niching method compares an offspring with some randomly chosen individuals from the current population. The most similar individual will be replaced if the offspring provides better fitness value. The crowding factor (*CF* parameter) is set to 2 or 3, which is used to control the size of the sample. The computational complexity is $O(NP)$. Replacement error is the main drawback of crowding. In addition, crowding is prone to clustering of individuals around the fittest basins.

4.1.2 Deterministic Crowding: To alleviate the problem of replacement error, Mahfoud introduced an improvement in original crowding, which independent of the parameter crowding factor. This method was named deterministic crowding [18]. It minimized the replacement error significantly and restored the selection pressure. However this method too has a drawback. It incurs loss of niches due to using tournament selection between similar individual within takeover time.

4.1.3 Probabilistic Crowding: To overcome the problem of missing local optima Mengshoel introduced probabilistic crowding [19]. A probabilistic replacement rule is used to maintain a high diversity in the population that replaces lower fitness individuals by higher fitness individuals according to their fitness proportion. The major problem of this procedure is that it has a very slow convergence rate and poor fine searching ability.

4.2 Crowding Differential Evolution

In CDE [2] an offspring is generated using the classical DE mentioned before. Then the Euclidean distances of the offspring with the other individuals are measured. The fitness of the offspring is compared with the most similar individual, i.e. the member which is nearest to the offspring (minimum Euclidean distance). The offspring will replace its most similar individual if and only if it provides better fitness value.

***Algorithm* 2**: Crowding Differential Evolution

1 Initialize population P and evaluate fitness fit_p.
2 ***while*** *termination condition* is not met **do**
3 ***for*** $i=1: NP$
4 Generate offspring vector U_i using basic DE and evaluate its fitness value $fit_U(i)$.
5 $p_r \leftarrow \min\left\{\left\|p_i, U_i\right\|\right\} \forall p_i \in P$ (most similar)
6 Replace p_r by U_i if its fitness is greater than equal to that of U_i. Update the fitness value.
7 ***endfor***
8 ***endwhile***.

However, the main drawback of CDE is poor fine searching ability and loss of niches because in this algorithm, the parent and its offspring find different optimal solutions.

5. OUR PROPOSED APPROACH: ClDES

In this part we will discuss the proposed algorithm in different subsections and at the end of this section a pseudo will be given for a clearer view.

5.1 CDE with locally informative mutation

In this proposed approach mutation phase is performed by picking random individuals from a selective population set rather than the whole population. This set of population can be termed as local neighborhood. It is determined by means of probabilistic selection which is described below. Let us consider there are NP real parameter D dimensional vectors in the population and $\vec{X}_i^{G}$ is the i^{th} individual of G^{th} generation. The probability of taking j^{th} member for performing the mutation is calculated as-

$$p(i,j) = 1 - \frac{\left\|\vec{X}_i^{G}, \vec{X}_j^{G}\right\|}{\sum_{k=1}^{NP}\left\|\vec{X}_i^{G}, \vec{X}_k^{G}\right\|} \quad (5)$$

where ||a,b|| is the Euclidean distance between a and b. Now based on these probability values, k individuals (generally set to 10 percent of the population size) are selected corresponding to each of the individual. The more the probability value more are its chances of being selected as a neighbor. Now from the *k-best* neighborhood three individuals are randomly chosen and mutation is performed according to (2). From this mutant vector offspring is generated applying Eq (3). Here the scale factor F and the crossover rate Cr are set to 0.9 and 0.1 respectively for the stabilization of niches and to maintain the diversity throughout the

population. Then selection is done between the offspring and most similar individual of the offspring in the entire population.

5.2 Detecting Environmental Changes

An efficient dynamic optimizer must detect the change of environment effectively. Here in this algorithm change detection in environment can be done by a *test solution*. At the very beginning of the algorithm a *test solution* is placed which does not take part in the optimization process. But at every generation the fitness value of the *test solution* is calculated. Whenever there is a change in the fitness value from the previous generation, it can be said that a change of environment has taken place and the algorithm needs to take necessary actions (including updating of the test solutions old fitness by new one).

***Algorithm* 3**: Speciation-Based Memory Archive: Reinitialization Of Population After A Change In Environment

Input: Population of the previous environment

1 Sort all the individuals in the population in the descending order of their fitness values.
2 ***for*** *i =1: no_of_species*
3 Set the best unprocessed individual as species seed.
4 Find the nearest *m* individuals of the species seed and set them as one species corresponding to the above mentioned species seed.
5 Remove processed members from the current population.
6 Randomly pick *m/2* individuals from the m individuals and reinitialize them within the search space.
7 ***endfor***

Output: Population for the newly changed environment.

5.3 Speciation-based memory archive

In most of the DOPs, the new environment is similar to the old one to a certain extent. So it is very natural that an algorithm will perform better after detecting an environment change if good solutions can be preserved in a memory archive. Now to handle the difficulties for locating multiple peaks a speciation based memory archive is introduced. In the event of change detection, the proposed algorithm does not reinitialize the entire population. Some of the population members are conserved for the following environment. Now which one should we conserve? Keeping in mind that environment is both dynamic and multimodal, algorithm have to store the set of the individuals in the memory archive. It is constituted by members representing the local or the global optima in the previous landscape. Hence for storage of the individuals, the whole population is partitioned into a number of species. In each species there must be a species seed. Now let us consider there is *m* number of individuals in each of the species with a species seed which provide maximum fitness value within a stipulated species. No species radius is taken for partition. Rather the nearest neighborhood (see Algorithm 4) concept is used. For simplicity we have taken *m* same as *k*. In each of the species, the species seed and half of the individuals taken randomly are kept unchanged in the archive for the following environment and the rest are reinitialized randomly in the search space. The combination of half newly generated member and half species-based individuals from the past environment constitutes the new population. The detailed algorithmic construction of the speciation-based memory archive can be given by *Algorithm* 3.

***Algorithm* 4**: ClDES Algorithm

1 Initialize a population $\overrightarrow{X}$ consisting of *NP* individuals in the search space randomly. Every individual is *D* dimensional.
2 Insert a *test solution* in the search space and evaluate it.
3 $k \leftarrow NP/10$, T←1.
while ! *termination condition( )* ***do***
for *i = 1:NP*
4 Calculate all the probability values p (i, j) for all j = 1, 2, …, *NP* and ($j \neq i$) according to (5).
5 Create a subpopulation of k individuals which provides maximum probability value corresponding to the i^{th} member.
6 Generate a mutant vector by (2) using three mutually exclusive individuals randomly from the subpopulation.
7 Create an offspring $\overrightarrow{U}_i$ by (3).
8 Determine the most similar (nearest by means of Euclidean distance) individual of $\overrightarrow{U}_i$ in the population $\overrightarrow{X}$.
9 Compare the fitness of the offspring to its most similar individual; replace the most similar individual if the offspring provides better fitness value.
endfor
// Detection of Environment Change
10 Evaluate the fitness of the *test solution*.
if f(test solution)$_{\mathbf{T}} \neq f$(test solution)$_{\mathbf{T-1}}$
11 Update the test solutions fitness.
12 Regenerate the population by ***Algorithm 3***.
endif
13 $T \leftarrow T+1$
endwhile.

6. EXPERIMENTAL SETUP

6.1 Moving Peaks Benchmark Problem

Branke proposed Moving Peaks Benchmark (MPB) in [11]. It has been widely used as a benchmark problem for DOPs. In this paper the authors have used MPB as the DOP of choice. In the MPB problem, several peaks can change with environment with respect to three features: location, width and height. For a *D* dimensional space the problem was designed as:

$$F\left(\overrightarrow{X},t\right)=\max_{1,2..N}\frac{H_i(t)}{1+W_i(t)\sum_{j=1}^{D}\left|x_j(t)-X_{ij}(t)\right|^2} \quad (6)$$

where W_i(t) and H_i(t) are the width and height of the i^{th} peak respectively at time t and X_{ij}(t) is the location of the j^{th} dimension of the i^{th} peak at time t. When the environment change occurs, the peak positions change in random direction of length S (shift length). It determines the challenge induced by the problem. The shifting of a single peak can be written as:

$$\vec{v}_i\left(t\right)=\frac{s}{\left|\vec{r}+\vec{v}_i(t-1)\right|}\left(\lambda\vec{v}_i(t-1)+\left(1-\lambda\right)\vec{r}\right) \quad (7)$$

λ is set to 0 to make the peak movements uncorrelated. The change in the attributes of a single peak can be given as:

$$H_i\left(t\right)=H_i\left(t-1\right)+height\_severity*\sigma \quad (8)$$

$$W_i\left(t\right)=W_i\left(t-1\right)+width\_severity*\sigma \quad (9)$$

$$\overrightarrow{X}_i\left(t\right)=\overrightarrow{X}_i\left(t-1\right)+\vec{v}_i\left(t\right) \quad (10)$$

where σ is a normally distributed random with zero mean and unity standard deviance. Initially heights of all the peaks are same

(denoted by I). The environment remains same for a certain number of fitness evaluations; after that different attributes of the peaks change following the rules described above. If the change frequency is E, then after every E number of function evaluations the environmental changes occur. The settings (both default and user-defined) used in this paper for tackling the challenge of detecting multiple optima are given in Table I.

Table I. Default and user-defined parameter settings used in our algorithm.

Parameter	Default Value	Values used in this paper
Number of peaks, p	10	{5, 10, 15, 20, 25}
Search Range	$[0,100]^D$	$[0,100]^D$
Shift Length, S	1.0	{1,2,3,4,5,6}
Dimension, D	5	{2, 3, 4, 5,10}
Correlation coefficient, λ	0	{0,0.1,0.3,0.5,0.7,1}
H	[30, 70]	[30, 70]
W	[1, 12]	[1, 12]
Height Severity	7.0	7.0
Width Severity	1.0	1.0
Initial Height, I	50	50
Change Frequency, E	5000	1000*D*(p/10)

6.2 Algorithms Compared

The performance of the proposed algorithm has been compared with two peer algorithms: Dynamic Species-Based Particle Swarm Optimizer (DSPSO) [14] and Clustering Particle Swarm Optimizer (CPSO) [9].

6.3 Performance Criteria

For judgment the performance of competing algorithms we utilize two performance metrics.

1: A*verage number of peak found*: This is a widely used metric for evaluation static niching algorithms and denotes the mean peak count over all independent runs. For dynamic problems, this metric refers to the mean value of average peak count over all independent runs. The average peak count in a single run refers to the sum of the individual optima count over all change instances divided by total number of instances.

2: Another metric used for comparison in is *Success Rate*. Its calculation is identical to the previous metric. Only difference is it is expressed in percentage of total runs in which all the optima (both global and local) were successfully detected.

All algorithms are executed using MATLAB 7.5 on Intel i5-760 machine with 2 GB RAM and 2.76 GHz speed. For conducting our experiments we have used 20 environmental changes in a single run and 50 such runs were simulated.

6.4 Parametric Setup

For creating homogenous experimental conditions, all the compared algorithms were allotted same initial number of trial solutions and made to run for same number of FEs (see Table II). Other control parameters for CPSO [9] and DSPSO [14] are fixed according to their literature. A threshold level of accuracy is kept fixed for every dimension D ($\varepsilon = 0.1*D$). As soon as the change is detected, if a solution's proximity to the peak is less than the threshold ε, it can be said that a peak is found. It is identical for all the algorithms. Detailed parametric setup is elucidated in Table II. Here $D = \{2, 3, 4, 5, 10\}$. With variation of D, ε and the change of frequency will vary accordingly as shown in Table II.

Table II. Detailed Parametric Setup.

Number of peaks, p	Initial number of trial solution	ε	Change Frequency (E)
5	100	0.1*D	500*D
10	200		1000*D
15	300		1500*D
20	400		2000*D
25	500		2500*D

7. RESULTS AND ANALYSIS

For analysis different comparisons have been undertaken using various configurations as reported discretely in Table I. First problem dimensionality (D) and the number of peaks are varied (p). Other attributes are set to default. Performance of different algorithms has been reported in Table III.

Table III. Average number peak found of Algorithms with change in D and p. (Best entries are marked in boldface)

No. of peaks (p)	Dimension (D)	ClDES	CPSO	DSPSO
5	2	**5**	**5**	**5**
	3	**5**	**5**	4.92
	4	**5**	3.81	3.66
	5	**5**	3.72	3.5
	10	**5**	3.2	2.96
10	2	**10**	**10**	**9.46**
	3	**10**	8.36	7.61
	4	**10**	7.84	7.13
	5	**9.31**	7.52	6.85
	10	**8.83**	6.32	5.05
15	2	**15**	11.8	10.2
	3	**14.91**	10.95	10.03
	4	**14.85**	10.62	9.82
	5	**14.9**	10.11	8.97
	10	**13.3**	7.5	4.67
20	2	**20**	15.67	14.23
	3	**19.97**	13.22	12.52
	4	**18.64**	11.16	9.2
	5	**18.52**	10.87	8.26
	10	**17.88**	9.55	6.26
25	2	**25**	20.3	17.25
	3	**24.62**	18.25	16.88
	4	**24.1**	17.8	15.2
	5	**22.36**	16.93	13.95
	10	**21.03**	12.35	10.13

From Table III it is evident that ClDES has outperformed CPSO and DSPSO except for the first case where all the algorithms detect every peak and in the second case where CPSO detects all the peaks while SPSO marginally fails to do so. The differences in the average number of peak found become significant when the number of peaks and the problem dimensionality increases. In spite of having greater search range in high dimensions, ClDES provide satisfactory results, unhindered by dimensional change. This is attributed to the fine searching ability of ClDES imparted

by local mutation. When the number of peaks increases, the speciation archive helps in tracking the optimal solutions.

Next we analyze Table IV that reports the comparison of different algorithms when the shift severity and the number of peak is varied keeping default value for all other problem parameters. Shift severity (*S*) is varied from 1 to 6 in steps of 1. The average number of peaks found for each case is given.

Table IV. Average No. of peak found of different algorithms on MPB for different shift severity (*S*) and *p*. (Best entries are marked in boldface)

p	*S*	ClDES	CPSO	DSPSO
5	1	**5.00**	3.72	3.50
	2	**4.97**	3.65	3.27
	3	**4.95**	3.55	3.19
	4	4.5	**4.53**	2.94
	5	**4.52**	2.81	2.16
	6	**4.19**	2.13	1.95
10	1	**9.31**	7.52	6.85
	2	**9.2**	7.11	6.11
	3	**8.94**	6.56	6.03
	4	**8.38**	6.4	5.7
	5	**8.43**	6.12	5.43
	6	**8.09**	6.09	4.94
15	1	**14.9**	10.11	8.97
	2	**14.65**	9.39	8.31
	3	**14.5**	9.95	8.26
	4	**14.21**	9.23	8.1
	5	**14.06**	9.21	7.82
	6	**13.8**	8.95	7.3
20	1	**18.52**	10.87	8.26
	2	**18.13**	11.31	7.85
	3	**17.95**	9.93	7.76
	4	**17.64**	10.06	7.2
	5	**17.21**	8.67	6.78
	6	**17.1**	7.13	6.25
25	1	**22.36**	16.93	13.95
	2	**20.35**	16.25	12.88
	3	**19.38**	15.31	10.58
	4	**18.22**	14.94	9.55
	5	**18.03**	13.83	7.26
	6	**17.53**	13.69	7.01

The tendency of algorithms to deteriorate with increasing shift severity is observed. Increasing shift severity results in greater displacement of the peaks relative to its previous position. Even so ClDES reports better performance than DSPSO and CPSO. It is interesting to note the maximum change in the number of peaks detected with peak severity changing from 1 to 6. Although marked differences do not exist, but keeping in mind the average peak count, which is more for ClDES, we can say that our algorithm is more consistent in the face of changing severity. DSPSO's use of radius parameter results in its poor performance when severities are greater. CPSO uses clustering for this framework but not with much success.

In Table V, the effect of the variation of λ has been studied. λ is the correlation coefficient which correlates the previous peak location with the present one using (7). With varying number of peaks (*p*) the average number of peaks found for different algorithms is given. The strength of the speciation based response is strongly demonstrated here. As the value of λ is increased the performance improves due to stronger correlation of present landscape with the previous one. Although CPSO and DSPSO perform fairly well in this setup, ClDES peak detection ability takes a strong stand and on an average it is much better than the other two. When speciation based response is used, historic data relating to previous landscape is available in the form the species individuals present in regenerated population. Once the peaks have been detected, this information goes a long away with the changing instances. The controlled diversity (local mutation) combined with speciation contributes to overall superiority.

Table V. Average No. of peak found of different algorithms on MPB for different *λ* and *p*. (Best entries are marked in boldface)

p	*λ*	ClDES	CPSO	DSPSO
5	0	**5**	3.72	3.5
	0.1	**5**	3.79	3.61
	0.3	**5**	3.92	3.78
	0.5	**5**	4.02	3.89
	0.7	**5**	4.3	3.92
	1	**5**	4.65	4.26
10	0	**9.31**	7.52	6.85
	0.1	**9.37**	7.54	6.89
	0.3	**9.5**	7.69	6.95
	0.5	**9.67**	7.78	7.09
	0.7	**9.88**	7.86	7.21
	1	**9.98**	7.91	7.66
15	0	**14.9**	10.11	8.97
	0.1	**14.92**	10.3	9.05
	0.3	**14.95**	10.61	9.22
	0.5	**14.97**	10.79	9.48
	0.7	**15**	10.94	9.67
	1	**15**	11.23	9.84
20	0	**18.52**	10.87	8.26
	0.1	**18.66**	10.62	8.35
	0.3	**18.7**	11.21	8.44
	0.5	**19**	11.54	8.51
	0.7	**19.36**	13.52	9.31
	1	**19.89**	14.11	9.42
25	0	**22.36**	16.93	13.95
	0.1	**22.94**	16.97	14.65
	0.3	**23.21**	17.51	15.71
	0.5	**23.38**	18.21	15.88
	0.7	**24.1**	19.32	16.81
	1	**24.69**	19.89	17.93

Table VI. Average Number of peak found by different algorithms on MPB for different change frequencies (*E*). (Best entries are marked in boldface)

E	ClDES	CPSO	DSPSO
2000	**9.06**	6.34	5.61
3000	**9.16**	6.89	6.32
4000	**9.29**	7.88	6.89
5000	**9.31**	8.52	7.85
6000	**9.36**	8.91	8.12

In Table VI we observe the ability of competing algorithms when the FEs budget (frequency *E*) is varied keeping default values for other parameters. In all the cases our algorithm has outperformed

the other two in terms of average number of peak found when a given accuracy level is used. This can be attributed to the strong optimizing tendency of DE. This proves that ClDES algorithm has very high convergence speed and can perform well with low FEs.

Table VII. Average number of peak found with variation in *p* using Speciation-based Memory and complete random initialization. (Best entries are marked in boldface)

p	Speciation-based memory archive	Complete random Initialization
5	**5**	3.21
10	**9.31**	7.53
15	**14.9**	11.06
20	**18.52**	12.58
25	**22.36**	15.67

To understand the utility of speciation-based memory archive experimentation is presented in Table VII. Average number of peak found by our algorithm with random reinitialization and with the speciation-based memory archive has been reported. From Table VII it can be seen that because of using speciation-based memory archive our algorithm gets a marked improvement over the random reinitialization technique. This shows the importance of memory usage in DOPs. For simple cases, as in low dimension search space where less number of peaks present and can be detected, we report the success rates achieved by all algorithms.

For this 2*D* and 3*D* MPB problems are used and the number of peaks used is {5, 10, 15}. The corresponding success rate (success means all the peaks are detected) is given in Table VIII. It is very clear from the results obtained that ClDES is a robust algorithm the number of missing peak is negligible.

Table VIII. Success rates at *D=2* and *D=3* with number of peaks *p=5* and *p=10*. (Best entries are marked in boldface)

D	*p*	ClDES	CPSO	DSPSO
2	5	**100**	**100**	**100**
	10	**100**	**100**	88
	15	**100**	20	12
3	5	**100**	**100**	96
	10	**100**	54	12
	15	**96**	0	0

For a clearer view, in Figure 1, population distribution at different instances is shown. Here a 2*D* MPB problem landscape is considered with 5 peaks. For visual clarity of the movement of peaks (marked in red), the shift length is set to 10. We have considered first three instances of a run. For every change instance 3 distributions have been shown in the figure with the gap of 500 function evaluations. In these diagrams red points correspond to the peak locations and the blue points correspond to the population individual.

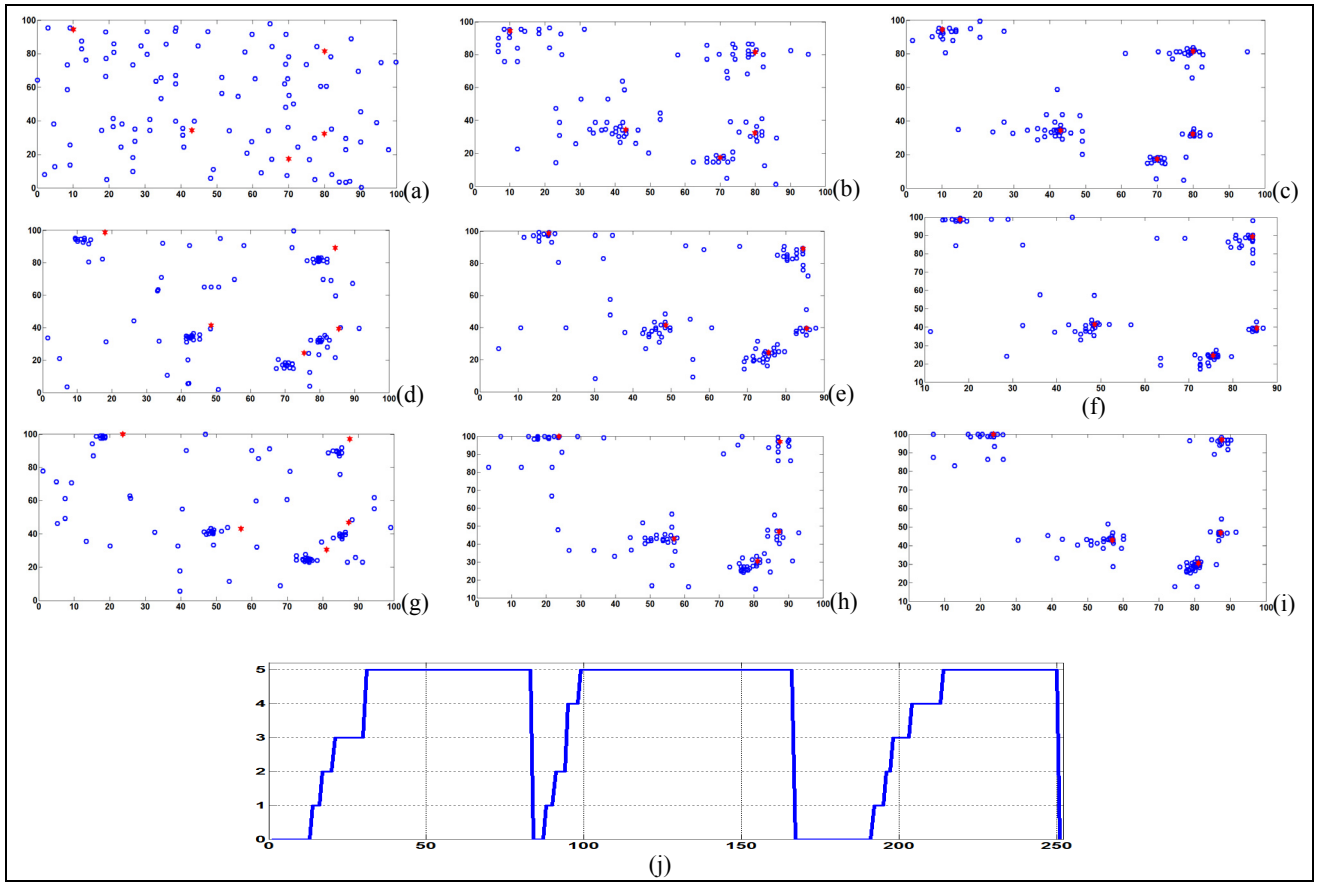

Figure 1. Population distribution at different FEs- (a) at FE=0, (b) FE=500, (c) FE=1000, (d) FE=1001, (e) FE=1500, (f)

Fig. 1 (a) indicates the initialization of the population. Fig 1 (c) is the final population of first change instance. In Figure 1 (d), using species based memory archive the population reinitialization after first change detection is shown. Similarly Figure 1 (f) is the final population of the 2nd change instance and Figure 1 (g) is regenerated population, using species-based memory archive, after 2nd change. Figure 1 (i) is the final population distribution at the end of 3rd change instance. It is clear from the figure that when the population is reinitialized using this newly introduced memory archive technique better solutions are kept which helps ClDEs to easily detect the peaks at the next instance. Even local mutation helps in reducing the gap between the vector and the offspring reducing the pressure of global optima. This is evident from the population distribution around different peaks of varying height (fitness value). Variation of average number of peak found with generation for the above mentioned case is shown in Figure 1 (j), which shows maximum peak detection at every instance after very few generations.

Based on the extensive experimental study we can say that ClDES is a robust optimizer that maintains consistency in its performance when put up against varied landscape features. The synergism of the individual components, local mutations and speciation-based archive, play a crucial role in establishing ClDES as a novel niching algorithm.

8. COCLUSIONS

This research work helps to establish a niching algorithm suited to dynamically changing landscape that utilizes a locally restricted mutation and speciation based memory archive to detect multiple peaks in non-stationary environments. The perceived advantage of the algorithm is its independency of niching parameters that greatly restricts the performance of EAs.

The hybridized ClDES technique takes a step forward in extending the concept of parameter-free algorithm in uncertain environments. Although many niching algorithms coexist, their application is restricted to functioning in static landscapes. It will be interesting to carry out a future investigation into their performance when a dynamic response feature, like the speciation based response stated in this work, is integrated.

9. REFERENCES

[1] R. Storn and K. V. Price. Differential evolution-A simple and efficient heuristic for global optimization over continuous Spaces. *Journal of Global Optimization*. 11, 4 (Dec. 1995), 341-359, 1995.

[2] R. Thomsen. Multimodal optimization using Crowding-based differential evolution. In *Proceedings of the Congress on Evolutionary Computation*. (19-23 June 2004) CEC '04. IEEE, 2, 1382-1389, 2004.

[3] X. Li. Efficient differential evolution using speciation for multimodal function optimization. In *Proceedings of the Conference on genetic and evolutionary computation.* (Washington DC, USA, 2005) GECCO '05. ACM, 873-880.

[4] R. Mendes and A. S. Mohais. DynDE: A differential evolution for dynamic optimization problems. In *Proceedings of the Congress on Evolutionary Computation.* (2-5 Sept. 2005) CEC' 05. IEEE, 2, 2808-2815, 2005.

[5] R. Angira and A. Santosh. Optimization of dynamic systems: A trigonometric differential evolution approach. *Computers & Chemical Engineering.* 31(9): 1055-1063, 2007.

[6] E. L. Yu and P. N. Suganthan. Evolutionary Programming with Ensemble of Explicit Memories for Dynamic Optimization. In *Proceedings of the Congress on Evolutionary Computation* (Trondheim, Norway, May 18-21, 2009). CEC '09. IEEE, 431-438.

[7] B-.Y. Qu, P. N. Suganthan, and J. J. Liang. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Trans. on Evol. Comput.* 16(5): 601 – 614, Oct. 2012.

[8] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, andV. Zumer. Dynamic Optimization using Self-Adaptive Differential Evolution. In *Proceedings of the Congress on Evolutionary Computation* (Trondheim, Norway, May 18-21, 2009). CEC '09. IEEE. 415-422.

[9] S. Yang, and C. Li. A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments. *IEEE Trans. on Evol. Comput.* 14(6): 959 – 974, Dec. 2010.

[10] Y. Jin and J. Branke. Evolutionary Optimization in Uncertain Environments—A Survey. *IEEE Trans. on Evol. Comput.* 9, 3 (June 2010), 303 – 317.

[11] J. Branke. Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems. *Proceedings of the Congress on Evolutionary Computation* (Washington DC, USA, July 6-9, 1999). CEC '09. IEEE, 3, 1875-1882.

[12] S. Yang. Explicit Memory Schemes for Evolutionary Algorithms in Dynamic Environments. *Evolutionary Computation in Dynamic and Uncertain Environments*. 51: 3-28, 2007.

[13] S. Yang and X. Yao. Population-Based Incremental Learning with Associative Memory for Dynamic Environments. *IEEE Trans. on Evol. Comput.* 12 (5): 542 – 561, Oct. 2008.

[14] D. Parrott and X. Li. Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation. *IEEE Trans. on Evol. Comput.* 10(4): 440 – 458, August, 2006.

[15] T. M. Blackwell and J. Branke. Multi-swarm Optimization in dynamic environments. In *Proc Appl. Evol. Comput.: EvoWorkshops 2004: EvoBIO, EvoCOMNET, EvoHOT, EvoISaP, EvoMUSART and EvoSTOC* (Coimbra, Portugal, 2004). LNCS. Springer, Heidelberg, Germany. Vol. 3005, 489-500, 2004.

[16] R. K. Ursem. Multimodal Gas: Multimodal optimization techniques in dynamic environments. In *Proc. of the Genetic and Evolutionary Computation Conference* (Las Vegas, Nevada, USA, July 8-12, 2000). GECCO '00. 19–26.

[17] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems*. Doctoral Thesis. University of Michigan. 1975.

[18] S. Mahfoud. *Niching Methods for Genetic Algorithms*. Doctoral Dissertation. University of Illinois, Urbana, Illinois, USA, 1995.

[19] O. Mengsheol and D. Goldberg. Probabilistic crowding: deterministic crowding with probabilistic replacement. *In Proc. of the Genetic and Evolutionary Computation Conference* (Orlando, FL, USA, July 13-17, 1999) GECCO '99. 1: 409–416.