Enhancing Clearing-based Niching Method Using Delaunay Triangulation

Shivam Kalra*, Shahryar Rahnamayan, SMIEEE† and Kalyanmoy Deb, FIEEE‡
*Department of System Design Engineering
University of Waterloo, Canada
Email: shivam.kalra@uwaterloo.ca
†Department of Electrical, Computer and Software Engineering
University of Ontario Institute of Technology, Canada
Email: shahryar.rahnamayan@uoit.ca
‡Department of Electrical and Computer Engineering
Michigan State University, USA
Email: kdeb@msu.com

plain [L]978-1-5090-4601-0/17/$31.00 ©2017 IEEE

***Abstract*—The interest in multi-modal optimization methods is increasing in the recent years since many of real-world optimization problems have multiple/many optima and decision makers prefer to find all of them. Multiple global/local peaks create difficulties for optimization algorithms. In this context, *niching* is well-known and widely used technique for finding multiple solutions in multi-modal optimization. One commonly used niching technique in evolutionary algorithms is the *Clearing* method. However, canonical clearing scheme reduces the exploration capacity of the evolutionary algorithms. In this paper, *Delaunay Triangulation based Clearing (DT-Clearing)* procedure is proposed to handle multi-modal optimizations more efficiently while preserving simplicity of canonical clearing approach. In DT-Clearing, cleared individuals are reallocated in the biggest empty spaces formed within the search space which are determined through Delaunay Triangulation. The reallocation of cleared individuals discourages wasting of the resources and allows better exploration of the landscape. The algorithm also uses an external memory, an archive of the explored niches, thus preventing the redundant visiting of the individuals, henceforth finding more solutions in lesser number of generations. The method is tested using multi-modal benchmark problems proposed for the *IEEE CEC 2013, Special Session on Niching Methods for Multimodal Optimization*. Our method obtains promising results in comparison with the canonical clearing and demonstrates to be a competitive niching algorithm.**

I. Introduction

Evolutionary Algorithms (EAs) are typically devised for converging to a single solution because of the globally employed selection scheme. However, most of the real-world problems exhibit the property of having more than one satisfactory solution. Process that involves finding multiple viable solutions for a given optimization problem is called *multi-modal optimization*. In particular, such problems are of great abundance in science and engineering viz. aerodynamic design, construction, scheduling, time/cost/resource scheduling. Multiple solutions for a given optimization problems are conceptualized as *peaks* (or *troughs* if minimization problem is assumed) and are often referred to as *niches*.

It is usually desirable to find multiple feasible solutions as it enables decision makers or experts to select the most appropriate solution depending of the domain and constraints of the problem. Inspired by the ecosystems in nature, *niching methods* are commonly employed for tasks dealing with multi-modal optimization. Niching is a generic term referred to as technique of locating and preserving the stable peaks/niches, or any potential candidates for optimum during the EA process.

All ecosystems have many different physical spaces (niches) with a finite amount of resources, which are apt for different inter-competing species. For example, on Earth, organisms living on land have different characteristics than organisms living in water bodies, allowing each of these groups of organisms to evolve independently within their respective niches. Thus, the ecosystem encourages the diversity and allows the preservation of various dissimilar species within their respective niches.

In context of optimization algorithms, niching methods are also inspired by nature, enabling to split the population into distinct sub-populations (niches) searching certain areas of the search space [1]. A niching method can be embedded into a standard EA to promote and maintain formation of multiple stable sub-populations within a single population, with the goal of finding multiple globally optimal or sub-optimal solutions. In a scenario of EA, the fitness symbolizes the resources of the niches and species are individuals grouped according to certain criteria (vicinity, fitness value and etc.), while niche corresponds to an optimum of the fitness landscape.

Evolutionary algorithms are well-established as strong candidates for tackling uni-modal optimization problems. However in multi-modal domain, many challenges exist; for instance, most niching techniques are not efficient in solving a multi-modal problem of a relatively large scale or with large number of optima, or detection of niches with variable radius or peak values. In addition, drastic limitations on their computation complexity still persist. Therefore, this field is an active area of research concerning development of niching strategies for EA as to benefit from their synergy for efficiently handling multi-modality.

Several niching methods have been developed previously, such as, stretching and deflation [2], crowding [3], fitness sharing [4], deterministic crowding [5], restricted tournament selection [6], clearing [7] etc. The comparisons have shown that clearing methods are efficient in reducing the genetic drift and maintaining multiple stable solutions [8]. However in the canonical clearing approach, cleared individuals have no chance to participate in the mutation and crossover, which limits the exploration capabilities of evolutionary process.

In multi-modal optimization domain, two criteria are generally used to measure the success of the search algorithms. First, whether an optimization algorithm can find all desired global/local optima within reasonable amount of time, and the second, if is capable of stably maintaining multiple candidate solutions. Clearing method for niching is successful in achieving the latter, however it falls short in former criterion. We are interested not only in stably identifying one or more global optima but we are interested to locate set of all acceptable solutions in timely manner.

In this paper, we present a novel multi-modal optimization algorithm based on an enhanced clearing procedure, we call it *Delaunay Triangulation Based Clearing (DT-Clearing)*. In this algorithm, cleared individuals are reallocated in the center of the large empty hyper-spheres formed within the search space. The proposed algorithm uses Delaunay Triangulation technique to find the large empty hyper-sphere. The proposed approach allows the non-elitist search by reallocation of cleared individuals, it is able to form stable niches across different local neighborhoods and eventually locates multiple global/local optima in lesser number of generations when compared with canonical clearing method.

The remainder of this paper is organized as follows. Section II provides brief problem statement for multi-modal optimization. Section III presents background and related work. Detailed explanation of algorithm is presented in Section IV and Section V covers results of our algorithm against benchmark test suite proposed for the *IEEE CEC 2013, Special Session on Niching Methods for Multimodal Optimization* and comparison with canonical clearing method and modified clearing approach. The conclusion remarks are presented in Section VI.

II. Problem Statement

The general aim of multi-modal optimization is similar to that of the standard optimization task, that is, in a given search domain $\mathcal{X}$, we seek to maximize/minimize

$$f(x), \quad x \in \mathcal{X} \tag{1}$$

For this paper, $\mathcal{X}$ is subjected to *box-constraint*, ie. any given $x \in \mathcal{X}$ follows

$$x_l \leq x \leq x_u \tag{2}$$

Where x_l and x_u are lower and upper bounds of x.

In the case of multi-modal optimization, we seek to find multiple $x^* \in \mathcal{X}$ that satisfies the constraint in Eq 2 and attains the maximum possible value of $f(x^*)$ in vicinity of x^* (local maxima). Therefore a given multi-modal optimization algorithms must successfully identify and maintain all the $x*$.

III. Niching Methods: A Brief Review

Simple Genetic Algorithm (SGA) is designed to converge at single solution; therefore in their classical form they are not useful in context of solving multi-modal problems. This limitation of SGA can be overcome by a mechanism of niching which allows GA process to identify multiple solutions. The single convergence characteristic of GA occurs because of the *Genetic Drift* which is an artifact of stochastic selection process used in GA with finite chromosome in the population. A naive niching method would be running SGA several times on the same problem and due to the stochastic nature of GA, one could attain multiple different solutions. However, most niching methods involve modifying the GA operators to allow formation and identification of multiple niches/solutions. One of the earliest niching method is Cavicchio's *Pre-selection Operator* [9] which was generalized to *Crowding* by De Jong [10] whereby the individuals produced by crossover and mutation of parents, replaces the most similar parent if it has a better fitness. Mengshoel in [5] made modifications to Crowding to reduce replacement errors, restore selection pressure, and remove the parameter called crowding factor (CF), thus resulting in the *Probabilistic Crowding*.

Another popular niching method is *Fitness Sharing* [4]; whereby fitness of an individual is lowered if there are many other similar individuals similar to it thus forcing GA to maintain diversity within the population. Fitness sharing technique has several drawbacks, e.g. it depends on the values of two parameters (the niche radius and the scaling factor), which cannot be easily determined. As a consequence, more advanced fitness sharing methods have been created such as implicit fitness sharing and co-evolution, where the fitness value of each agent depends on the fitness values of the remaining individuals in the population [11], as well as the Dynamic Niche Sharing (DNS) and the Dynamic Fitness Sharing (DFS) [12], both aiming at the self identification of the niches in the population. In [13], authors have used fuzzy logic to determine the niche radius to achieve the Dynamic Niche Sharing (DNS).

In the last couple of decades, many niching methods have been proposed, popular ones are – Fitness Sharing [4], Deterministic Crowding [14], Probabilistic Crowding [5], Clustering [15], Restricted Tournament Selection [6] and Clearing [7]. Most niching algorithms perform defectively when the dimensionality of the problem or the number of optima increases, and some schemes cannot successfully preserve the previously found solutions. The problem of retention of found optima points can be solved by archiving procedure. Archiving in multi-modal optimization, allows preserving all the potential optima candidates in efficient data structure for fast retrieval, search and insertion. In [16], Epitropakis et al. have used dynamic archiving algorithm which adapts its archive radius dynamically. In [17], authors Ellaban et al. have implemented efficient archiving to keep it free of duplicates and fast retrieval.

A comparison of these multi-modal optimization methods in [8] has shown that clearing is simple yet efficient method for forming stable multiple niches in GA and are generally more efficient for exploration of the problem search space. Clearing method for niching was introduced by Petrowski in [7] and showed that clearing method can effectively succeed in reducing the effect of genetic drift. In Clearing sub-populations/niches are identified based on certain similarity measures, such as Euclidean distance. Eventually, only most fit individual is kept in each sub-population by setting fitness of all other individuals to zero. Throughout this paper, we refer to this original clearing as canonical clearing method. Unlike other niching methods where resources are shared among similar individuals, clearing methods prefer to distribute resources only among elitists. However, clearing is very simple and intuitive niching technique but there are few major drawbacks: **1)** cleared or individuals with low fitness values take very high penalty and do not participate in GA while taking population slots, **2)** cleared individuals limit the exploration capabilities of search space, **3)** parameter such as niching radius is usually dependent on problem landscape and hard to predetermine. Over past years, various modification has been proposed to canonical clearing method to enhance its capabilities further. In [18], Imrani et. al combines sharing and a fuzzy clustering technique to improve the performance of genetic algorithms in solving multi-modal problems. Singh and Deb in [8] proposed a modified clearing approach, where cleared individuals are re-allocated outside of their basin of attractions or sub-population in search of finding unknown optima instead of wasting the population slots. In their modified clearing approach, cleared individuals are relocated within $1.5 \times \sigma_{clear} - 3.0 \times \sigma_{clear}$ from their basin of attractions; where σ_{clear} is niching radius.

In [17], Ellaban and Ong have introduced Valley-Adaptive Clearing schema which consists of three core phases: **1)** *valley identification*: which assigns group of individuals to a particular valley and dominant individual of each valley is archived, which eventually becomes solution points, **2)** *valley clearing phase*: weak individual belong to particular valley is reallocated away from all other valleys randomly based on niche radius for that particular valley, **3)** *Valley replacement phase*: individuals of populations belonging to previous encountered valleys are replaced with individuals in new basin of attractions in order to promote the search of more unexplored regions in the landscape. In [19], Magda et al. proposed another Clearing scheme called Context Based Clear (CBC) which uses homogeneity of sub-population to decide whether all or only nearest neighbors should be cleared, henceforth it regulates the radius size of the area cleared around the pivot of sub-populations. CBC approach is shown to converge to solutions faster than canonical clearing method [19]. In [20], authors use idea of clearing in the local sharing for efficient multi-modal optimization.

IV. Delaunay Triangulation Based Clearing

Delaunay Triangulation based Clearing (DT-Clearing) is clearing procedure that makes use of the knowledge about big empty spaces within the population and subsequently reallocate the cleared individuals to these locations for a better exploration of landscape and faster convergence. The proposed approach comprise of the following two main component: **1)** canonical clearing, and **2)** Delaunay Triangulation for finding empty spaces.

A. Canonical Clearing Niching Technique

Clearing is a niching procedure suggested by Petrowski, which is inspired by the principle of sharing of limited resources within sub-populations/niches of individuals grouped based on certain factors [7].

Every niche has a dominant (master) individual, i.e. the one with the best fitness within that niche. An individual fall in to the same niche if its Euclidean distance to the master is less than a given threshold known as the clearing radius (σ). The method shares the resources of a niche among a fixed number of winners (κ) (individuals with top fitness values within a given niche), while individuals with lower fitness value within same niche are set to zero-fitness value (minimum fitness). 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 candidate solutions having a critical distance measure (σ 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 unchanged. After the clearing is over, the candidate solutions closer to the next best κ solutions are cleared as before and the fitness values of these next best κ solutions are kept as is. This procedure is continued till all candidate solutions are considered. The pseudo-code for the canonical clearing based niching is given in Algorithm 1.

The clearing procedure is shown to efficiently reduce the genetic drift when used with an appropriate selection operator [7]. However, at line 16 of Algorithm 1, fitness value of inferior individuals are set to minimum which restrict them from participating in crossover or mutation operations during the next generations. These cleared individuals take the population slots without having to contribute towards evolution which hinders the exploration capacity of the entire evolutionary algorithm.

Therefore, we propose a modification to the clearing procedure whereby, instead of wasting the population slots, we move the cleared individuals to *big empty and unexplored areas* within search space, in an attempt to find the better solutions and subsequently form the niches/sub-populations within those areas.

Our proposed approach uses *Delaunay Triangulation* to locate the empty regions within the search space ($\mathbb{R}^n$) as explained in the next section.

Algorithm 1 Pseudo-code for canonical clearing-based niche formation technique

```
 1: procedure CLEARING($P_p$)                ▷ Where $P_p$ is population
 2:     $P_p \leftarrow Sort(P_p)$   ▷ Sort the population $P_p$ in decreasing order of their fitness values
 3:     for $i \in [0, S-1]$ do
 4:         if $Fitness(P_p[i] = -\infty)$ then
 5:             Continue
 6:         end if
 7:         $nbWinners \leftarrow 1$
 8:         for $j \in [i+1, S-1]$ do
 9:             if $Fitness(P_p[j]) = -\infty$ then
10:                 continue
11:             end if
12:             if $Distance(P_p[i], P_p[j]) < \sigma$ then
13:                 if $nbWinners < \kappa$ then
14:                     $nbWinners \leftarrow nbWinners + 1$
15:                 else
16:                     $Fitness(P_p) \leftarrow -\infty$
17:                 end if
18:             end if
19:         end for
20:     end for
21: end procedure
```

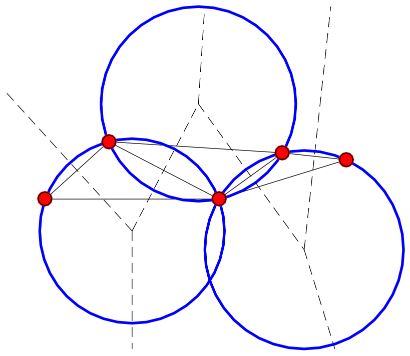


Fig. 1: Visual depiction of *circumcircle property* of Delaunay triangulations

B. Delaunay Triangulation

Delaunay triangulation for a set P of points in a plane is a triangulation $DT(P)$ such that no point in P is inside the *circumcircle* of any triangle in $DT(P)$ as shown in Figure 1. Here "triangulation" is extended from the planar usage to arbitrary dimension.

An important property of Delaunay Triangulations is *Circumcircle property* which states that any circumcircle formed by any of the triangulations in Delaunay Triangulations is empty as shown by empty blue circles in Figure 1.

Circumcircle property of Delaunay Triangulation is used to find the empty circles, which are sorted by their radius to find largest empty regions within the search space. For the proposed method, Delaunay Triangulation algorithm packaged with the standard *Python's Scipy* library is used. Algorithm used for Delaunay triangulation works with set of P where $P \in \mathbb{R}^n$ and $n \geq 2$.

C. Proposed Algorithm

After doing the clearing, as described in Algorithm 1, we search for the cleared individuals which are identified with fitness value of $-\infty$. Thereafter, each of these individuals are moved around the center of large empty hyper-spheres (empty circles for 2D space) formed within the population. Empty hyper-spheres are constructed using circumcircle property of Delaunay Triangulation.

The proposed clearing approach is called before *selection* operation of GA. This approach has all the strengths of clearing method. In addition, the reallocation of inferior individuals allows the method to explore empty regions of the search space for better solutions and eventually forming more niches. By placing the individuals around/to the center of large empty hyper-spheres also ensuring that individuals are not placed within other niches thus not disrupting the niching formations.

Pseudo-code for the proposed method is depicted in Algorithm 2. The algorithm is the direct replacement of *clear* method from the evolutionary algorithm's main loop. There are the following two important components within Algorithm 2:

- **CalculateVolumeHS (line 6)**: Estimating the volume enclosed in the intersection of hyper-spheres and search-space.
- **ReallocateInd (line 8)**: Reallocating strategy of cleared individuals around the center of large empty hyper-spheres

1) Estimate of the volume enclosed in intersection of Hyper-Sphere HS_i and Search Space: Since, empty hyper-spheres represent the empty regions within the search space, it is apparent that algorithm tries to move the cleared individuals to the largest ones first.

Algorithm 3 Pseudo-code for finding the estimate for effective volume V_e of hyper-sphere HS_i

```
 1: procedure ESTIMATE-EFFECTIVE-VOLUME($HS$)
 2:     $C \leftarrow Center(HS)$
 3:     $r \leftarrow Radius(HS)$
 4:     $C_{max} = C + r$
 5:     $C_{min} = C - r$
 6:     $Ret = 1$                        ▷ Initial estimated volume
 7:     for $i \in [0, n)$ do              ▷ $n$ is cardinality of $C$
 8:         $L_{max} \leftarrow \min C_{max}[i] X_{max}[i]$
 9:         $L_{min} \leftarrow \max C_{min}[i] X_{min}[i]$
10:         $Ret \leftarrow Ret \times (L_{max} - L_{min})$
11:     end for
12:     Return $Ret$
13: end procedure
```

Since search space is constrained to box-constraint i.e. x_l, x_u. Therefore, sometimes, hyper-spheres formed from the

Algorithm 2 Pseudo-code for Delaunay Triangulation based clearing (DT-Clearing) for niche formation

```
1: procedure DT_CLEARING($P_p$)                                    ▷ Where $P_p$ is population
2:     $Clear(P_p)$                                                ▷ Call the original Clear procedure (algorithm 1)
3:     $P_{if} \leftarrow FindFitnessInd(P_p, -\infty)$            ▷ Select individuals with fitness value of $-\infty$ as set by Clear
4:     $D \leftarrow CalculateDelaunayTriangulations(P_p)$         ▷ Calculate Delaunay Triangulation from current population
5:     $S \leftarrow FindCircumHyperSpheres(D)$                    ▷ Calculate circum-hyper-spheres from triangulations
6:     $VS \leftarrow CalculateVolumeHS(S)$                        ▷ Calculate effective estimated volume of all the empty spheres
7:     for $i \in [0, Len(P_{if})]$ do
8:         $ReallocateInd(P_{if}[i], VS)$                          ▷ Reallocate within empty hyper-spheres based on volumes $VS$
9:         $Fitness(P_{if}[i]) \leftarrow f(P_{if}[i])$            ▷ Refresh the fitness value after reallocation, $f$ is fitness function
10:    end for
11:    $Archive(P_p)$                                              ▷ Perform archiving of the best individuals in the population
12: end procedure
```

population intersect with the search-space's boundaries thus reducing the effective region allowed for reallocation.

Effective Volume V_e – Volume (area for 2D problems) of the space enclosed within intersection of empty hyper-sphere HS_i and the search space is called *Effective Volume* V_e of hyper-sphere HS_i.

In order to estimate the effective volume V_e, we use the volume of intersection between circum-hyper-rectangle of HS_i and search space as shown in Figure 2 (in 2D search space). The estimate plays nicely with our algorithm as requirement of the algorithm is to make only a relative comparison of absolute effective volume (V_e).

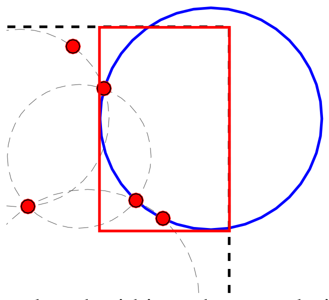


Fig. 2: Area enclosed within red rectangle is used as estimate for *Effective Volume* V_e; black dotted line represents the boundaries of search space & blue circle represents empty circle formed through Delaunay Triangulations.

The pseudo-code for calculating the estimate of effective volume (V_e) hyper-sphere (HS_i) is given in Algorithm 3. It takes hyper-sphere HS_i as input and calculates the Effective Volume (V_e) by considering the volume of the space formed by intersection of circum-hyper-rectangle of hyper-sphere (HS_i) and search-space.

2) Reallocation heuristics for the Cleared Individual (I_c): After volume estimation step, we reallocate the cleared individuals within empty hyper-spheres determined through Delaunay Triangulation. Reallocation is repeated until all of the cleared individuals are placed outside the niche of every candidate solution having a non-zero (or value-to-reach (VTR)) fitness.

There are two steps involved in reallocating of a given cleared Individual I_c, **1)** select an empty hyper-sphere for reallocation, **2)** choose a location within selected hyper-sphere.

We use heuristics for carrying out the steps defined above in order to achieve the goal of higher exploration and efficient niche formations while preserving stochastic nature of the algorithm.

Selecting the hyper-sphere for reallocation: All the empty hyper-spheres with their center lying within the existing niches are filtered out. Then effective Volume V_e of all the remaining empty hyper-sphere is calculated. The probability of an individual I_c to be reallocated within a hyper-sphere HS_i is proportional to its Effective Volume V_e.

$$HS = HS_1, HS_2.., HS_n \quad HS \text{ is set of hyper-sphere candidates for the reallocation of individual } I_c$$

$$V = V_1, V_2, .., V_n \quad V_i \text{ is effective volume of hyper-sphere } HS_i$$

$$P(HS_i) = \frac{V_i}{\sum_0^n V_i} \quad \text{Probability to select } HS_i$$

Choosing the location within the selected hyper-sphere for reallocating the Ic: Once the hyper-sphere HS_i is selected for reallocation of individual I_c, it is not always moved to the center, instead, its relocation position is chosen such that its distance from the center of hyper-sphere HS_i is normally distributed with mean μ of 0 and σ of $\frac{R}{3}$ (giving $\sim$ 99% confidence within $\pm R$ from center). A cleared individual I_c is reallocated in any random direction using unit vector r_u.

$$C = Center(HS_i) \quad \text{Center of } HS_i$$

$$R = Radius(HS_i) \quad \text{Radius of } HS_i$$

$$RN_n = randn(0, \frac{R}{3}) \quad \mu = 0 \text{ and } \sigma = \frac{R}{3}$$

$$r_u = randu(n) \quad \text{Random unit vector}$$

$$I_f = C + (r_u * (RN_n * R)) \quad \text{New location of } I_c$$

D. Archiving Procedure

All the optimum points found throughout the GA are stored in archive. Archive allows querying centers of all the niches formed and its associated solutions sorted according to their fitness values. Archive is implemented as hierarchical tree in Python using `scipy's cKDTree`. First level nodes of the tree represents the niche center and any node below them are the individuals belonging to that niche found during the entire history of GA. The nodes in archive tree are automatically merged based on the defined minimum archive radius which is kept around same as niching radius σ.

V. Experimental Study

The aim of multi-modal optimization is to locate multiple peaks/optima in a single run and to maintain these found optima until the end of a run. An algorithm for solving multi-modal optimization problems should have following abilities: **1)** nding maximum number of global/local optima, and **2)** preserve these found solutions until the end of algorithm.

A. Benchmark Overview

To evaluate performance of proposed DT_Clearing, we employed the CEC 2013 Benchmark for multi-modal optimization [21]. The benchmarks were ran on Sharcnet computing cluster with code written in Python programming language. The original benchmark contains 20 multi-modal test functions. First three multi-modal test functions in the benchmark suite were not used as they are 1D and our approach requires at minimum 2D search space.

- F_1 : [Not Used] Five-Uneven-Peak Trap (1D)
- F_2 : [Not Used] Equal Maxima (1D)
- F_3 : [Not Used] Uneven Decreasing Maxima (1D)
- F_4 : Himmelblau (2D)
- F_5 : Six-Hump Camel Back (2D)
- F_6 : Shubert (2D, 3D)
- F_7 : Vincent (2D, 3D)
- F_8 : Modified Rastrigin (2D)
- F_9 : Composite Function 1 (2D)
- F_{10} : Composite Function 2 (2D)
- F_{11} : Composite Function 3 (2D, 3D, 5D, 10D)
- F_{12} : Composition Function 4 (3D, 5D, 10D, 20D)

These multi-modal test functions are designed to have following properties [22]:

1) Designed as maximization problems.
2) F_6 to F_8 are scalable multi-modal functions. The number of global optima for F_6 and F_7 are determined by their dimension D. However, for F_8, the number of global optima is controlled by the user.
3) F_9 to F_{12} are scalable multi-modal functions. F_9 and F_{10} are separable, and non-symmetric, while F_{11} and F_{12} are non-separable, non-symmetric complex multi-modal functions. The number of global optima in all composition functions is independent from their dimension D, and therefore can be controlled by the user.

B. Performance Measures

The performance is measured in terms of the (i) *peak ratio* PR, (ii) *success rate* SR, and (iii) *averaged number of evaluations*, $AveFEs$, attained while locating all global optima over NR multiple runs achieving for a given accuracy ϵ.

Definition 1: Peak Ratio (PR) measures the average percentage of all known global optima found over NR runs and it is calculated by the following equation:

$$PR = \frac{\sum_{run=1}^{NR} NPF_i}{NKP * NR},$$

where $NPF_i =$ number of global found at i^{th} run and,

$NKP =$ number of known global optima

Definition 2: Success Rate (SR) calculates the percentage of successful runs out of all runs NR. A successful run is defined when all global optima are found, given by:

$$SR = \frac{NSR}{NR},$$

where $NSR =$ number of successful runs,

$NR =$ number of total runs

Definition 3: Convergence speed (AvgFEs) is calculated by counting the average number of function calls that are required to locate all optima for a given accuracy ϵ over NR runs and it is given as follows:

$$AveFEs = \frac{\sum_{run=1}^{NR} FE_i}{NR},$$

where $FE_i =$ number of functions calls at i^{th} run

C. Experiment Settings

The evolutionary algorithm using proposed clearing approach and canonical clearing approach were implemented and executed for multi-modal test problems with $NR = 50$ runs at each of the 5 levels of accuracy $\epsilon = \{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}\}$. The maximum allowed function evaluations $MaxFEs$ for each of the test problem is selected using following rule as given in [21]:

- 5×10^4 for F_4 to F_5 (2D)
- 2×10^4 for F_6 to F_{11} (2D)
- 4×10^4 for F_6 to F_{12} ($\geq$ 3D)

Each run finishes when number of function call reaches the $MaxFEs$. For every i^{th} run FE_i is calculated which represents the number of function evaluations required until all global optima were located.

Table I shows properties of test problems in CEC 2013 benchmark, where first column $Index$ is index of multi-modal function in original benchmark suite, $Function$ shows function name and dimension of the function, r is niche radius, $F*$ is value of global optima and last column NKP show number of global optima.

TABLE I: Properties of multimodal functions in CEC 2013 benchmark

Index	Function	r	F^*	NKP
4	$F_4(2D)$	0.01	200.0	4
5	$F_5(2D)$	0.5	1.03163	2
6	$F_6(2D)$	0.5	186.7309	18
7	$F_7(2D)$	0.2	1.0	36
8	$F_6(3D)$	0.5	2709.0935	81
9	$F_7(3D)$	0.2	1.0	216
10	$F_8(2D)$	0.01	-2.0	12
11	$F_9(2D)$	0.01	0.0	6
12	$F_{10}(2D)$	0.01	0.0	8
13	$F_{11}(2D)$	0.01	0.0	6
14	$F_{11}(3D)$	0.01	0.0	6
15	$F_{12}(3D)$	0.01	0.0	8
16	$F_{11}(5D)$	0.01	0.0	6
17	$F_{12}(5D)$	0.01	0.0	8
18	$F_{11}(10D)$	0.01	0.0	6
19	$F_{12}(10D)$	0.01	0.0	8
20	$F_{12}(20D)$	0.01	0.0	8

D. Results

Both the clearing schemes – proposed and standard are implemented over the base evolutionary algorithm with same parameters as given in Table II.

Locations of optima point found by standard clearing (left) and proposed clearing scheme (right) are shown in Figure 3 for various multi-modal functions taken at 200^{th} generation. It should be noted that red points are archived optima and white ones are optima found in generation. It is evident from Figure 3 that proposed method is much efficient in exploring the landscape. Figure 4 shows the error plot for one of the run of both the approaches on *Griewank* function, it can be seen that proposed approach converges much faster and with lesser variance.

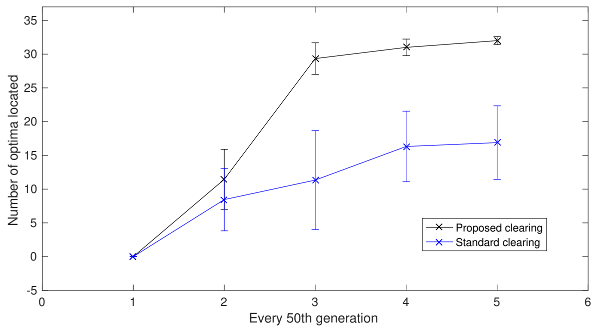


Fig. 4: Error plot for number of optima located every 50^{th} generation by proposed clearing vs standard clearing for *Griewank* function

Table III and IV show peak ratio PR and success ratio SR for proposed clearing approach and standard clearing approach. It can be seen that proposed algorithms is significantly better in exploring and maintaining the multiple solutions. From function index 4–10 are highly multi-modal, thus PR and SR are much higher for our proposed algorithms compared to the standard clearing approach. If the given archiving algorithm is able to sustain the found optima, then PR parameter intuitively gives information about how fast and how many optima are being discovered throughout the algorithms. The PR value of proposed algorithms are generally higher than PR values of standard clearing for all the test problems at all levels of accuracy. However, both the approaches have hard time in finding all the optima points highly multi-modal or very high dimensional problems, however standard clearing is not able to get even single *successful run* for $F_{12}(3D)$ however proposed clearing is able to get 11 successful runs at accuracy level of e^{-1}.

TABLE II: Parameters setting for used for running multi-modal test functions

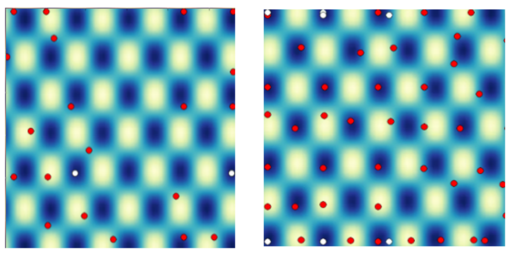
Parameters	Values
GA Specific Parameters	
Population Size	100
Mating	Two-point crossover
Mutation Function	Gaussian($\sigma = 0.1$)
Selection Function	Tournament($size = 3$)
Mutation Prob.	0.1
Crossover Prob	0.5
Clearing Specific Parameters	
Niche radius σ	Dependent on function
Winner per niche κ	1

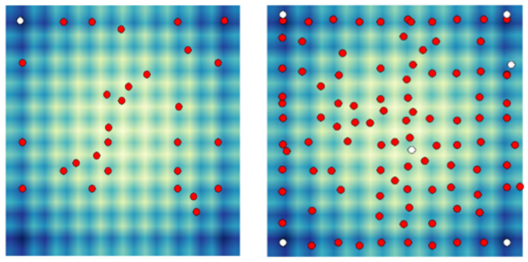
Converge speed $AvgFes$ for proposed clearing method stayed lower than standard clearing method. Since on most test problem both algorithms could not achieve successful runs, therefore most of time convergence speed is maximum allowed function evaluations $MaxFEs$. It can be seen from table III for $F_8(2D)$, SR values of modified approach is very high compared to canonical clearing, showing quick better superiority of finding/sustaining multiple solutions. Furthermore, high difference in PR values of highly multi-modal problem $F_7(3D)$ with 216 optima is also an indication of better landscape visibility of proposed clearing scheme.

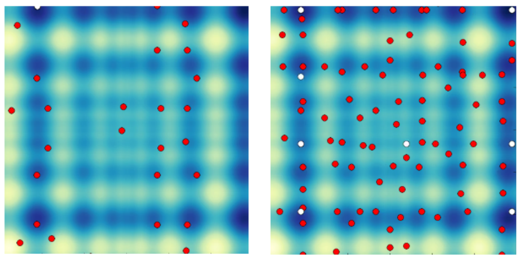
VI. Conclusion

In this paper, a new multi-modal optimization evolutionary search scheme – Delaunay Triangulation based Clearing *DT-Clearing* has been proposed which involves reallocating the cleared individuals within empty regions of population to enhance the canonical clearing procedure. In this study, we have also considered incorporating the archiving procedure with proposed as well as canonical clearing scheme.

Performance of our proposed algorithms was compared with standard clearing using multi-modal test problems. The results has shown that the proposed clearing scheme finds many more optima than standard clearing scheme at much faster pace. But, benefits of proposed method also comes with the extra computation effort (specially calculation of Delaunay Triangulations) during the evolutionary process. However both the clearing schemes require the predefined niche radius σ and κ parameters which are difficult to determine without knowledge of solution landscape. Delaunay Triangulation sometimes have


(a) Griewank Function


(b) Rastrigin Function


(c) Schwefel Function

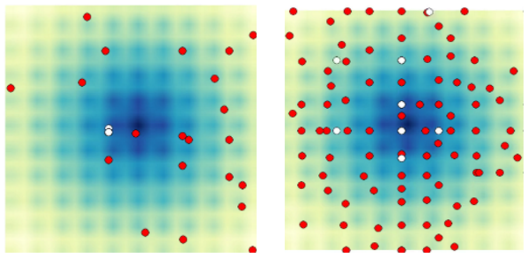

(d) Ackley Function

Fig. 3: Optima found using standard clearing (left) vs proposed clearing (right) using same archiving and base evolutionary algorithm for various multi-modal functions (white dots represents current generation solutions and red dots represents solutions within archive).

TABLE III: Peak ratios (PR) and success rate (SR) of Canonical vs Modified clearing approach for $F_4(2D)$ – $F_{11}(10D)$

Accuracy level ϵ	$F_4(2D)$				$F_5(2D)$				$F_6(2D)$				$F_7(2D)$				$F_6(3D)$			
	Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
$1.0e^{-1}$	0.725	0.480	**1.000**	**1.000**	1.000	0.980	1.000	0.980	0.230	0.000	**0.410**	0.000	**0.326**	0.000	0.210	0.000	0.123	0.000	**0.289**	0.000
$1.0e^{-2}$	0.610	0.500	**1.000**	**1.000**	1.000	0.900	1.000	0.900	0.133	0.000	**0.323**	0.000	0.230	0.000	**0.360**	0.000	0.071	0.000	**0.234**	0.000
$1.0e^{-3}$	0.540	0.440	**0.995**	**0.980**	1.000	0.860	1.000	**0.900**	0.110	0.000	**0.323**	0.000	0.153	0.000	**0.311**	0.000	0.061	0.000	**0.111**	0.000
$1.0e^{-4}$	0.480	0.440	**0.980**	**0.980**	0.725	0.200	**0.900**	**0.900**	0.090	0.000	**0.260**	0.000	0.152	0.000	**0.223**	0.000	0.051	0.000	**0.060**	0.000
$1.0e^{-5}$	0.310	0.420	**0.995**	**0.980**	**0.710**	0.200	0.680	**0.240**	0.020	0.000	**0.250**	0.000	0.080	0.000	**0.170**	0.000	0.001	0.000	**0.060**	0.000

Accuracy level ϵ	$F_7(3D)$				$F_8(2D)$				$F_9(2D)$				$F_{10}(2D)$				$F_{11}(2D)$			
	Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
$1.0e^{-1}$	0.089	0.000	**0.430**	0.000	0.739	0.440	**1.000**	**1.000**	0.725	0.000	**1.000**	**1.000**	0.431	0.000	**0.455**	**0.000**	0.625	0.100	**0.710**	**0.320**
$1.0e^{-2}$	0.071	0.000	**0.203**	0.000	0.810	0.000	**1.000**	**1.000**	0.610	0.000	**0.930**	**0.660**	0.433	0.000	0.433	**0.000**	0.611	0.000	**0.653**	**0.020**
$1.0e^{-3}$	0.034	0.000	**0.207**	0.000	0.833	0.000	**1.000**	**1.000**	0.567	0.000	**0.920**	**0.640**	0.322	0.000	**0.356**	**0.000**	0.609	0.000	**0.653**	0.000
$1.0e^{-4}$	0.031	0.000	**0.105**	0.000	0.650	0.000	**1.000**	**1.000**	0.563	0.000	**0.980**	**0.425**	0.311	0.000	**0.310**	**0.000**	0.567	0.000	**0.737**	0.000
$1.0e^{-5}$	0.001	0.000	**0.100**	0.000	0.750	0.000	**0.950**	**0.980**	0.534	0.000	**0.650**	**0.300**	0.133	0.000	**0.278**	**0.000**	0.156	0.000	**0.610**	0.000

Accuracy level ϵ	$F_{11}(3D)$				$F_{12}(3D)$				$F_{11}(5D)$				$F_{12}(5D)$				$F_{11}(10D)$			
	Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified		Canonical		Modified	
	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
$1.0e^{-1}$	0.510	0.020	**0.710**	**0.310**	0.683	0.000	**0.963**	**0.220**	0.22	0.000	**0.663**	0.000	0.000	0.000	**0.345**	0.000	0.000	0.000	0.123	0.000
$1.0e^{-2}$	0.483	0.000	**0.653**	**0.020**	0.578	0.000	**0.756**	0.000	0.220	0.000	**0.663**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-3}$	0.321	0.000	**0.567**	0.000	**0.571**	0.000	0.542	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-4}$	0.311	0.000	**0.333**	0.000	0.432	0.000	**0.525**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-5}$	0.311	0.000	**0.333**	0.000	0.160	0.000	**0.565**	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

TABLE IV: Peak ratios (PR) and success rate (SR) of Modified vs Canonical clearing approach for $F_{12}(10D)$ and $F_{12}(20D)$

Accuracy level ϵ	$F_{12}(10D)$				$F_{12}(20D)$			
	Canonical		Modified		Canonical		Modified	
	PR	SR	PR	SR	PR	SR	PR	SR
$1.0e^{-1}$	0.000	0.000	**0.020**	0.000	0.000	0.000	0.000	0.000
$1.0e^{-2}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-3}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-4}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
$1.0e^{-5}$	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

also failed to find the required triangulations satisfying the property of forming empty circles (hyper-sphere) within the search space, if individuals in the population are co-linear, in those cases then proposed approach fails to clear any individual. However in our experiments, it never occurred as all the problems are defined in real space. However, one could expect it to happen often in discrete or constraint search spaces.

For future extension of the proposed method, authors would like to incorporate **1)** adaptive niche radius using techniques, viz in [13, 23, 24], **2)** fast approach for calculating Delaunay Triangulation as in [25], **3)** dynamic archiving of best optima points, viz in [16] and **4)** performing local search around best solution of each niche (for higher accuracy) as in [26, 26, 27].

It can be concluded that idea of using Delaunay Triangulation to find empty regions to accelerate exploration capability of multi-modal optimization has a lot of potential to be a competitive technique. This the first research paper that has used Delaunay Triangulation combined with clearing to achieve superior niching, however authors feel it can be implemented across various optimization algorithms (such as CMA-ES) for further enhancing their capabilities. It is also intended to extend the application of *DT-Clearing* to various real life applications in order to test its performance. The idea proposed in this paper can be extended for constrained multi-modal optimization.

VII. Acknowledgment

The authors would like to thank *NSERC* for its support for this work through scholarship/grant. Authors would also like to thank *SHARCNET* for providing the computing resources required for this research. Finally, thanks to authors of open source libraries – *DEAP* [28] and *SCOOP* [29] for their contribution in making research tools open and accessible.

References

[1] X. Li et al. "Seeking Multiple Solutions: An Updated Survey on Niching Methods and Their Applications". In: *IEEE Transactions on Evolutionary Computation* PP.99 (2016), pp. 1–1.

[2] K. E. Parsopoulos et al. "Objective Function "stretching" to Alleviate Convergence to Local Minima". In: *Nonlinear Analysis: Theory, Methods & Applications*. Proceedings of the Third World Congress of Nonlinear Analysts 47.5 (Aug. 1, 2001), pp. 3419–3424.

[3] B. Sareni and L. Krahenbuhl. "Fitness Sharing and Niching Methods Revisited". In: *IEEE Transactions on Evolutionary Computation* 2.3 (Sept. 1998), pp. 97–106.

[4] David E. Goldberg and Jon Richardson. "Genetic Algorithms with Sharing for Multimodal Function Optimization". In: *Proceedings of the Second International Conference on Genetic Algorithms on Genetic Algorithms and Their Application*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1987, pp. 41–49.

[5] Ole Mengshoel and David Goldberg. "Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement". In: *Proc. of the Genetic and Evolutionary Computation Conference (GECCO-99)* (July 1, 1999), pp. 409–416.

[6] Georges R. Harik. "Finding Multimodal Solutions Using Restricted Tournament Selection." In: *ICGA*. 1995, pp. 24–31.

[7] A. Petrowski. "A Clearing Procedure as a Niching Method for Genetic Algorithms". In: *, Proceedings of IEEE International Conference on Evolutionary Computation, 1996.* , Proceedings of IEEE International Conference on Evolutionary Computation, 1996. May 1996, pp. 798–803.

[8] Gulshan Singh and Kalyanmoy Deb Dr. "Comparison of Multi-Modal Optimization Algorithms Based on Evolutionary Algorithms". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. GECCO '06. New York, NY, USA: ACM, 2006, pp. 1305–1312.

[9] DJ Cavicchio. "Adaptive Search Using Simulated Evolution". University of Michigan, 1970.

[10] K. DEJONG. "An Analysis of the Behavior of a Class of Genetic Adaptive Systems". In: *Ph. D. Thesis, University of Michigan* (1975).

[11] P. J. Darwen and Xin Yao. "Speciation as Automatic Categorical Modularization". In: *IEEE Transactions on Evolutionary Computation* 1.2 (July 1997), pp. 101–108.

[12] A. Della Cioppa, C. De Stefano, and A. Marcelli. "Where Are the Niches? Dynamic Fitness Sharing". In: *IEEE Transactions on Evolutionary Computation* 11.4 (Aug. 2007), pp. 453–465.

[13] J. Gan and K. Warwick. "Dynamic Niche Clustering: A Fuzzy Variable Radius Niching Technique for Multimodal Optimisation in GAs". In: *Proceedings of the 2001 Congress on Evolutionary Computation, 2001*. Proceedings of the 2001 Congress on Evolutionary Computation, 2001. Vol. 1. 2001, 215–222 vol. 1.

[14] Samir W. Mahfoud. "Niching Methods for Genetic Algorithms". In: *Urbana* 51.95001 (1995), pp. 62–94.

[15] Xiaodong Yin and Noël Germay. "A Fast Genetic Algorithm with Sharing Scheme Using Cluster Analysis Methods in Multimodal Function Optimization". In: *Artificial Neural Nets and Genetic Algorithms*. Ed. by Dr Rudolf F. Albrecht, Dr Colin R. Reeves, and Dr Nigel C. Steele. Springer Vienna, 1993, pp. 450–457.

[16] Michael G. Epitropakis, Xiaodong Li, and Edmund K. Burke. "A Dynamic Archive Niching Differential Evolution Algorithm for Multimodal Optimization". In: *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 79–86.

[17] Mostafa MH Ellabaan and Yew Soon Ong. "Valley-Adaptive Clearing Scheme for Multimodal Optimization Evolutionary Search". In: *Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on*. IEEE, 2009, pp. 1–6.

[18] A. El Imrani et al. “A Fuzzy Clustering-Based Niching Approach to Multimodal Function Optimization”. In: *Cognitive Systems Research* 1.2 (June 1, 2000), pp. 119–133.

[19] Magda B. Fayek, Nevin M. Darwish, and Mayada M. Ali. “Context Based Clearing Procedure: A Niching Method for Genetic Algorithms”. In: *Journal of Advanced Research* 1.4 (Oct. 2010), pp. 301–307.

[20] Grant Dick and Peter A. Whigham. “Weighted Local Sharing and Local Clearing for Multimodal Optimisation”. In: *Soft Computing* 15.9 (June 15, 2010), pp. 1707–1721.

[21] Xiaodong Li, Andries Engelbrecht, and Michael G. Epitropakis. “Benchmark Functions for CEC’2013 Special Session and Competition on Niching Methods for Multimodal Function Optimization”. In: *RMIT University, Evolutionary Computation and Machine Learning Group, Australia, Tech. Rep* (2013).

[22] M. W. Pereira, G. S. Neto, and M. Roisenberg. “A Topological Niching Covariance Matrix Adaptation for Multimodal Optimization”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. 2014 IEEE Congress on Evolutionary Computation (CEC). July 2014, pp. 2562–2569.

[23] Ofer M. Shir and Thomas Bäck. “Niche Radius Adaptation in the CMA-ES Niching Algorithm”. In: *Parallel Problem Solving from Nature - PPSN IX*. Ed. by Thomas Philip Runarsson et al. Lecture Notes in Computer Science 4193. Springer Berlin Heidelberg, 2006, pp. 142–151.

[24] Mike Preuss. “Niching the CMA-ES via Nearest-Better Clustering”. In: *Proceedings of the 12th Annual Conference Companion on Genetic and Evolutionary Computation*. GECCO ’10. New York, NY, USA: ACM, 2010, pp. 1711–1718.

[25] Ahmad Biniaz and Gholamhossein Dastghaibyfard. “A Faster Circle-Sweep Delaunay Triangulation Algorithm”. In: *Advances in Engineering Software* 43.1 (2012), pp. 1–13.

[26] B. Y. Qu, J. J. Liang, and P. N. Suganthan. “Niching Particle Swarm Optimization with Local Search for Multi-Modal Optimization”. In: *Information Sciences* 197 (Aug. 15, 2012), pp. 131–143.

[27] Dingcai Shen and Xuewen Xia. “A Local Search Particle Swarm Optimization with Dual Species Conservation for Multimodal Optimization”. In: *Information Computing and Applications*. Ed. by Baoxiang Liu, Maode Ma, and Jincai Chang. Lecture Notes in Computer Science 7473. Springer Berlin Heidelberg, Sept. 14, 2012, pp. 389–396.

[28] Félix-Antoine Fortin et al. “DEAP: Evolutionary Algorithms Made Easy”. In: *Journal of Machine Learning Research* 13 (Jul 2012), pp. 2171–2175.

[29] Yannick Hold-Geoffroy, Olivier Gagnon, and Marc Parizeau. “Once You SCOOP, No Need to Fork”. In: *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*. XSEDE ’14. New York, NY, USA: ACM, 2014, 60:1–60:8.