Extension of Δ_p SMS-EMOA for 3-D Benchmark Functions

Soumyadip Sengupta
Günter Rudolph

Algorithm Engineering Report
TR12-2-004
Sept. 2012
ISSN 1864-4503

Faculty of Computer Science
Algorithm Engineering (Ls11)
44221 Dortmund / Germany
http://ls11-www.cs.uni-dortmund.de/

Project Report

Extension of Δ_p SMS-EMOA for 3-D benchmark functions

Soumyadip Sengupta

Supervisor: Prof. Dr. Günter Rudolph

September 16, 2012

Contents

1 Introduction

This Project work is funded by DAAD WISE Programme and done as a part of Internship in TU Dortmund. My sincere thanks goes to Prof. Dr. Günter Rudolph, without whom this wouldn't have been possible.

In this project the main idea is to extend the already proposed Δ_p SMS-EMOA [7] to 3-D benchmark functions. The main idea of SMS-EMOA [2] was to select an individual over another by hyper-volume measure using mu+1 Evolution Strategy.But this does not lead to an evenly distributed Pareto front that we are looking for in the context of multiobjective online control [7]. Averaged Hausdorff distance measure [10] is another recently proposed performance metric that can be used with more accuracy than traditional Generational Distance or Inverse Generational Distance measure. Δ_p SMS-EMOA have been developed with an aim to incorporate Averaged Hausdorff distance measure to obtain an algorithm generating evenly distributed Pareto fronts. Here the algorithm maintains an archive, and solutions are updated depending on the Averaged Hausdorff distance measure. Now for computing this measure we need the knowledge of the reference set with which we shall compare the current approximated Pareto front. This reference set is formed by uniform distribution of the points on the current Pareto front. For 2-D this was easier, by building piece-wise linear approximation of the front and distributing points uniformly over it. With 3-D this job is very challenging and itself a good problem of computational geometry. Mainly the difficulty is that the points are not all on the same hyperplane. The idea is to construct a surface with pieces of interconnected triangles. This is called Triangulation. Delaunay's Triangulation is one such commonly used triangulation technique. Here after the triangulation is done on the set of points of the Approximated Pareto front we observe triangles of different areas being formed. While calculating the area of each triangle we neglect some triangles which contribute very little to the overall area, to speed up the run time. Now considering the minimum area of a triangle we go on diving each triangle into equal parts such that area of each part is nearly equal to the minimum area. Now we find that the range of the areas of the triangles is very small compared to the total area. We replace each triangle with their vertices. This forms an uniform distribution of points on the surface. And now we employ clustering over these huge set of points to produce required reference set points. Even after this though the points are nearly uniformly distributed, still the boundary has some imperfections. So to make the boundary uniform, we detect the boundary of the front and form a piece-wise linear approximation of it. Now we place those new boundary points over this piece-wise linear boundary at regular intervals. Thus we successfully produce the reference set. Now the second part of the project deals with writing the whole code of Δ_p SMS-EMOA in Matlab. The original updation strategy have been modified to some extent for better performance. We have simulated the algorithm on DTLZ test bed [4].

The article is classified into several sections which describe the project work more clearly. The sections are made in order of the work done. Section 2 deals with triangulation and formation of a surface from a set of points. Section 3 deals with division of the triangles into a number of smaller triangles of approximately equal area. Section 4 deals with the final part of construction of the reference set and use of clustering to form it. Section 5 outlines the update strategy used with SMS-EMOA along with the pseudo-code of the generation of reference set. Section 6 presents the result of simulation of the complete Δ_p SMS-EMOA algorithm over DTLZ1 and DTLZ2 test cases. And at last we conclude the article in Section 7.

2 Delaunay's triangulation

In computational geometry and computer graphics Delaunay's Triangulation is one of the most popular method for generation of a surface from a set of points. Mathematically it is defined as : a Delaunay Triangulation for a set of points P in a plane is a triangulation $DT(P)$ such that no point in P is inside the circumcircle of any triangle in $DT(P)$. The basic idea associated with it is to maximize the minimum angle of the triangle and thereby avoiding skinny triangles. There are several mathematical properties and discussions associated with this topic which is certainly beyond the scope of this project. There have been many algorithms [9] which have successfully implemented Delaunay's Triangulation. Some of the famous algorithms are Flip Algorithms; Divide and Conquer; Sweephull [5] etc. In this case we have used the already available Matlab code for Delaunay's Triangulation.

From a given set of points, which in this case is the present approximated Pareto front we build the triangulated surface with the help of Delaunay's Triangulation. Now the job is to place N_R points uniformly over this surface. Matlab have several inbuilt commands for executing delaunay's triangulation.

One such command is : TRI = delaunay(X,Y) ; creates a 2-D Delaunay triangulation of the points (X,Y), where X and Y are column-vectors. TRI is a matrix representing the set of triangles that make up the triangulation. The matrix is of size mtri-by-3, where mtri is the number of triangles. Each row of TRI specifies a triangle defined by indices with respect to the points. Then we can use triplot or trimesh to plot the figure.
Here we are actually dealing with 3-D cases. To build a surface we perform Delaunay's triangulation in x-y plane in 2-D and then raise the points to their respective heights to form the surface. One such sample case is shown in Figure 1. From an already given set of points, which is approximated Pareto front in 3-D for DTLZ2 we show the triangulation.

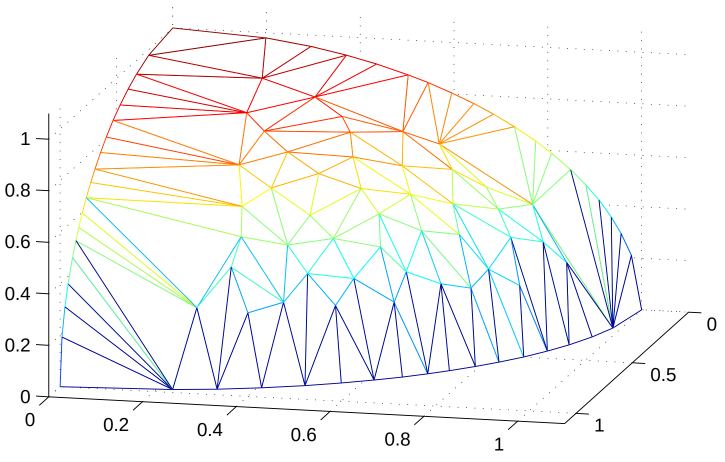

Figure 1: Delaunay's Triangulation for DTLZ2 test case

3 Division of triangles

Now as we have a set of triangles forming a surface, we observe the area of each triangle. Observing the areas for several different cases we infer that there are some triangles which are very small in area compared to the average area contribution of each triangle and deletion of this triangle won't effect the overall surface area. Now the real importance of this deletion will be clear when we discuss the principle of division of the triangles. Now of the remaining triangles we choose the triangle with the least area as the standard and aim to divide every triangle equally into triangles of area same as the least area. In this way we will have triangles of equal area and by collecting them we can make N_R polygons of equal area. But in practice this is very hard to realize. Dividing the triangles into n parts where n is in the order of 2 is easier. But in this process we can't keep the area of each part of the triangle equal to the least area of all the triangles. Still the area of each small pieces of triangles of one larger triangle are all equal and approximately equal to the required least area. The standard deviation of the area triangles formed is very small. So practically we can assume that we have divided the whole surface into large numbers of small triangles of equal area. In Figure 2 we show the boxplot of the area of the triangles produced to have an idea regarding the variation. Now for collecting the triangles to make N_R uniform points, it is better to at first replace the set of triangles with their respective vertices instead of centroids as it reduces the number of points and helps the computation. Also selecting the centroids makes the net boundary of the refernce set move more inward which is not intended.

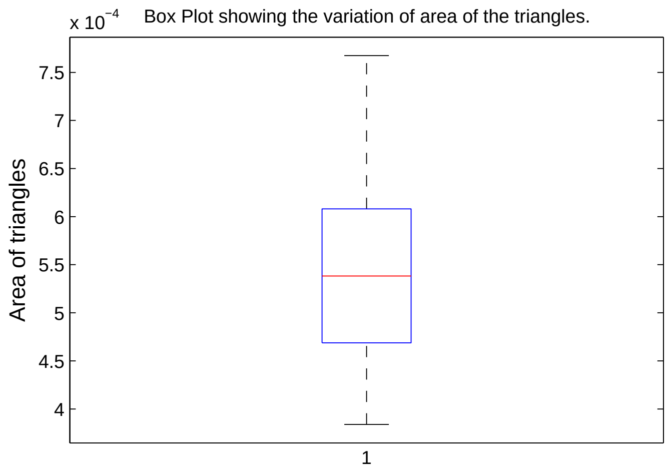

Figure 2: Boxplot of the area of the triangles produced

The logic behind the division of the triangle is simple. We divide a triangle by drawing median over the longest side. In this way we avoid making skinny triangles. A recursive routine is needed which would continuously divide a triangle into several equal parts until we have the required number of triangles $N = 2^n = \lceil \frac{total\ area\ of\ surface}{required\ least\ area} \rceil$, where n is the number of steps of recursive division need. Algorithm 3 represents the pseudocode for the division of a triangle operation.

Figure 3 shows the intermediate step where the surface is divided into equal number of small triangles and Figure 4 shows the next step where the triangles are replaced with their respective centroids for sample DTLZ2 test case. From the experiment it is noted that approximately around 3,000 triangles are formed for DTLZ2 test case after division of triangles, with 5% below mean area as the deletion criteria and around 1200 triangles for 10% deletion criteria. One interesting thing is that for DTLZ1 there is hardly any variation in number of triangles formed with both 5% and 10% deletion criteria and in both case is around 1,250. Deletion criteria above 10% is not recommended as it hampers the uniformity.

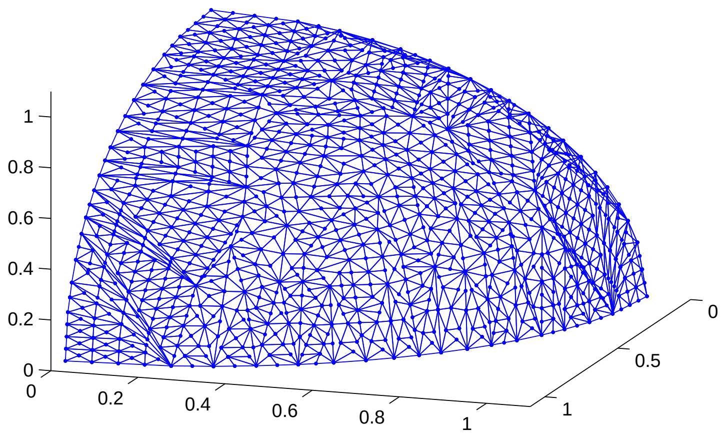

Figure 3: Division of the surface into smaller triangles of equal area

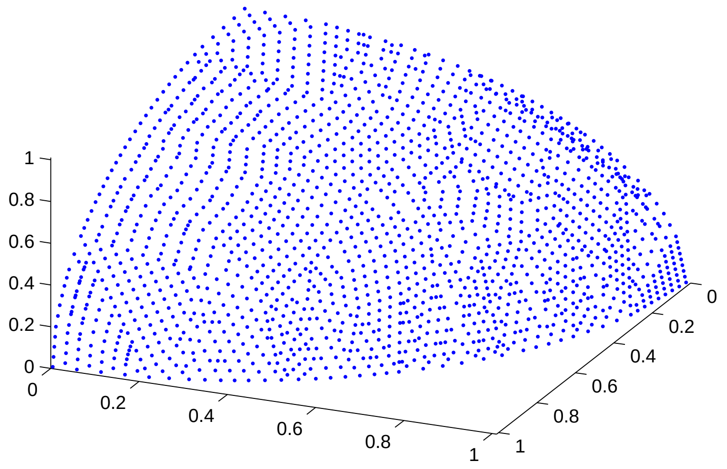

Figure 4: Replacement of triangles with their centroids

4 Clustering and creation of reference set

Now one of the easiest way to collect these vertices to form equi-distribution of data points is to perform clustering. But before we perform clustering it is better to perform a boundary adjustment procedure to organise the rough boundary.

4.1 Boundary Adjustment

There are two possible ways of boundary adjustment. One method is α-hull method and other is edge checking method. The former one is more time-consuming but produces accurate results. There are some problems associate with edge checking method and we are working to resolve it. If this works well, then it would be much faster than α-hull method.

4.1.1 α-hull method

For boundary adjustment, the first job is to detect the border. For this we perform multi-dimensional scaling and transform the data set into 2-D plane from 3-D. MDS (Multi-Dimensional Scaling) [6] operates on a given matrix of dissimilarities of distance $d(i,j)$ between the points. The mapping is performed in such a way that the distances $d(i,j)$ in the original space are maintained as closely as possible by solving the optimization problem:

$$min \sum_{i,j=0;i\neq j}^{n} (\|z_i - z_j\| - d(i,j))^2.$$

The determination of boundary of a set of points in 2-D space is a difficult task. Convex hull is one such method for shape detection but it is very difficult to detect boundary for any set of points which are not strictly convex in nature. From a statistical perspective the problem is reflected by the distribution based estimation of an unknown set S given a random sample R^S of points from S. For this purpose we use the theory of α-convex hulls [1] which is coded in Matlab. Basically, the concept of convex hulls is generalized. A set S is α-convex if any point in the complement S^c of S is separable from S using an open ball of radius α. The α-convex hull then is defined as the smallest α-convex set which contains R^S. By varying the parameter a the degree of considered concavity can be controlled.It becomes obvious that the α- convex hull converges to the convex hull with increasing α.

From the given set of already formed collection of vertices of the triangles, we perform MDS and convert the data-set to 3-D. Over this data set we apply alpha-hull method to determine the bound of the data set. Now after the boundary points are detected, we build a piece-wise linear approximation of the boundary by joining consecutive points on the boundary. The total length of this piece-wise linear bound is calculated and dividing it by total number of boundary points we get the required distance between two consecutive points on the boundary. And now maintaining this required distance we adjust the positions of the point on the boundary. Thus the job of forming the modified boundary is complete. Boundary adjustment is a very delicate thing. At the start of the run there is no need to use boundary adjustment as it adds very little to the overall result. Algorithm 4 shows the boundary adjustment routine over the original reference set.

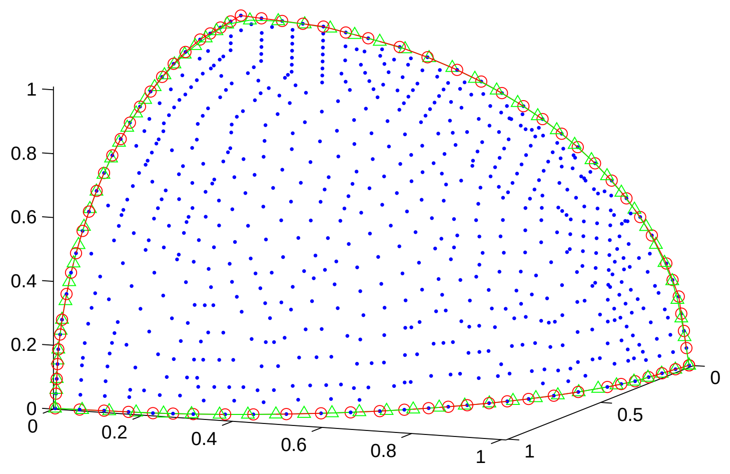

Figure 5: Boundary adjustment of the reference set

Figure 5 shows the boundary adjustment procedure with the new boundary and the old one.The red dot shows the old boundary and the green triangles show the boundary after adjustment. The blue dots represent the remaining triangle vertices which have been plotted for 20% area cut-off threshold for better representation.

4.1.2 Edge Checking Algorithm

The basic idea is to produce a faster edge detection algorithm rather than the traditional α-hull method. Here from the set of points we first construct a triangulation of the surface. Now for each node in the triangulation we compute the connectiong nodes for each such node. Now for each edge we compute the intersection of the set of connecting nodes of the two nodes comprising the edge. If the intersection produces only 1 node, then the edge belongs to the border. But in this case, due to some problem in the triangulation, there are situations where the algorithm exactly detects two borders of the Pareto surface but only detects partially the other border. We are presently working to resolve this problem.

Algorithm 5 describes the edge checking algorithm for border detection. After the boundary is detected the same method of boundary re-adjustment through linear interpolation model as described before is done.

4.2 Clustering

Clustering algorithm like Fuzzy C-means [3] or K-means [8] serve the purpose very well. Clustering based on Euclidean distance will produce a nearly uniform distribution. Figure 6 shows the clustering part and the placement of cluster centroids among the triangle centroids. The blue dots are the centroids of the triangle and the red dots are the cluster centroids or the reference points. Matlab have inbuilt code for both of them.

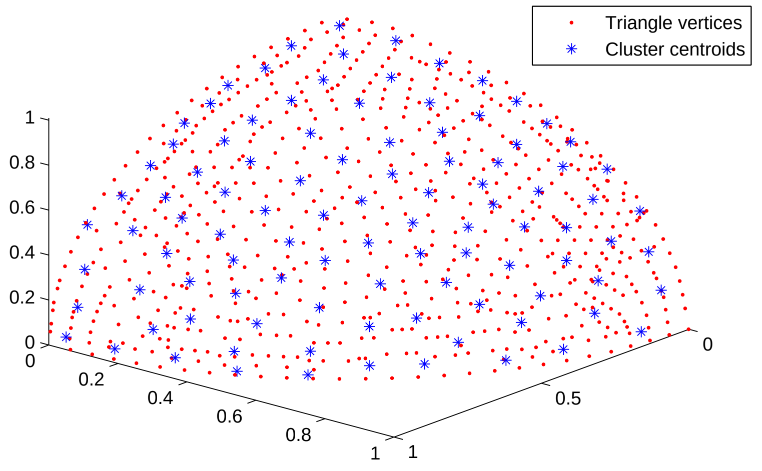

Figure 6: Clustering of triangle centroids to produce reference set

[IDX, C] = kmeans(X, K) partitions the points in the N-by-P data matrix X into K clusters. This partition minimizes the sum, over all clusters, of the within-cluster sums of point-to-cluster-centroid distances. Rows of X correspond to points, columns correspond to variables. It returns the K cluster centroid locations in the K-by-P matrix C. There is also option for passing the initial positions of the cluster centroids into the algorithm. For details please refer to the Matlab manual.

But in Clustering the Cluster centroids are not on the original surface constructed but in somewhere else. This is theoretically a wrong concept. There may be two ways to circumvent this problem. One is to place the Cluster centroid to a point on the nearest triangle plane. This is more accurate process but quite a time consuming one. As the triangles themselves are very small in area, we place the Cluster centroid to it's nearest point in the cluster group. In that way the final reference points remains the same as that of the previous case. Figure 7 shows the Original Centroids and Final Centroid after Shifting to the nearest point in the Cluster group.

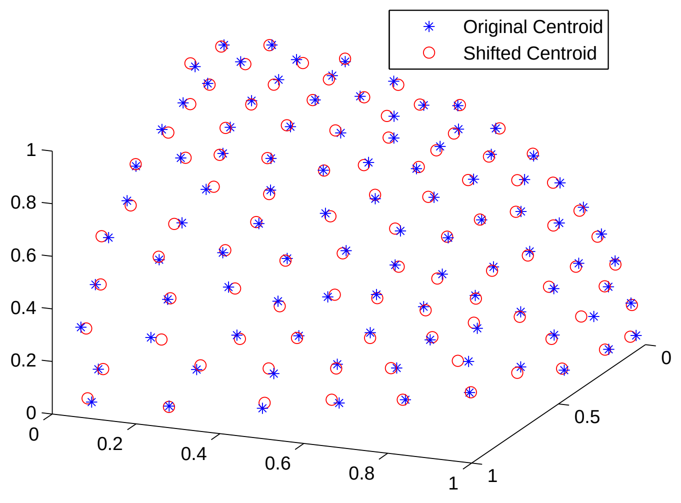

Figure 7: Shifting of Cluster Centroids

Finally Figure 8 shows the ultimate reference set formed after the complete procedure for DTLZ2 and Figure 9 shows the same for DTLZ1. Algorithm 2 is the routine for the generation of reference set.

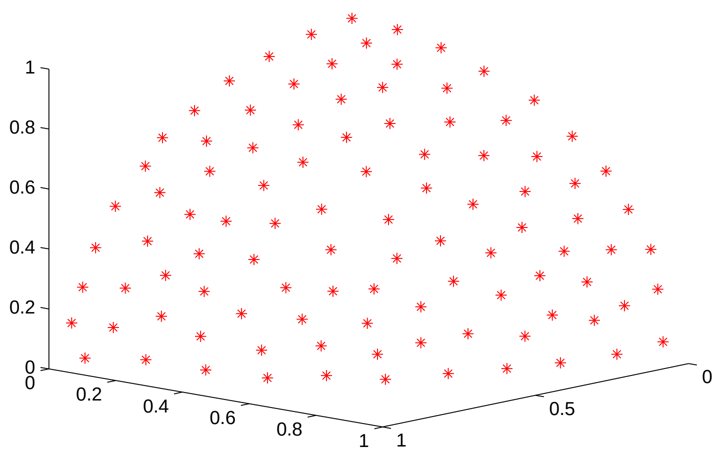

Figure 8: Final reference front for DTLZ2

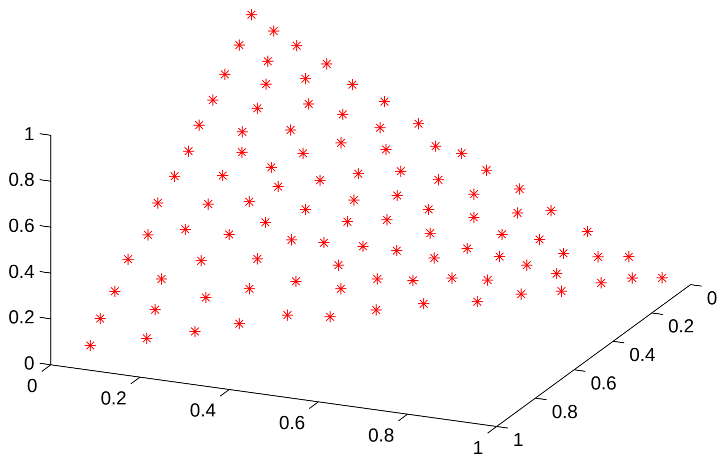

Figure 9: Final reference front for DTLZ1

5 Net archival update strategy

The update strategy is almost similar to that of Δ_p SMS-EMOA in 2-D, except the fact that we have used the generation of reference front only when it is needed, i.e. only when the Achieve size crosses $N_R = 100$ so that we need reference set to compute Averaged Hausdorff distance measure. Also we observe that in the update often is the case that more than one solution have the lowest corresponding Averaged Hausdorff distance measure, though their corresponding GD and IGD contribution may not be both same. To circumnavigate the problem we employ another measure for only those solutions with minimum Average Hausdorff distance value. Here we reject the solution having the closest neighboring solution in the archive. The rest of the strategy is same as before. Algorithm 1 shows the main updation strategy of the Δ_p SMS-EMOA in 3-D.

Algorithm 1 update(x,A,P) ; Updation Strategy of Δ_p SMS-EMOA in 3-D

```
$A = M_f\ (A \cup \{x\}, \leq)$.
if card(A)> $N_A$ then
    $R$ =generate_ref(P).
    for all a ∈ A do do
        $h(a) = \Delta_1(A \backslash \{a\}, R)$.
    end for
    $A^*$ =argmin $\{h(a) : a \in A\}$.
    if card($A^*$)> 1 then
        for all i ∈ $A^*$ do do
            $c(i) = GD(\{i\}, A)$.
        end for
        $i^*$ =argmin $\{c(i) : i \in A^*\}$.
    else
        $i^* = A^*$
    end if
    $A = A \backslash \{i^*\}$.
end if
return A
```

Algorithm 2 generate_ref(P) ; Generation of reference set from current Pareto front

```
tri=delaunay(P).
for each $i \in tri$ do
    area(i)=area of each i
end for
eliminate triangles in $tri$ with $area(i) < 0.05 * \bar{area}$.
for each $i \in tri$ do
    $K = \lceil \log_2\{\frac{area(i)}{min(area)}\} \rceil$.
    $newtri$ =divide(tri(i),K).
end for
$points$ =set of vertices of all $newtri$.
new_points=boundary_adjustment($points$).
$C = Kmeans(new\_points, N_R)$.
Replace each C with the nearest point in it's own cluster group to form
Refernce Set R
return R
```

Algorithm 3 divide(T,K); Division of each triangle into K parts

```
$k = 1$.
while $k \neq K + 1$ do
  for $i := 1 \rightarrow 2^{k-1}$ do
    Divide T into 2 equal parts by drawing median on the longest side.
  end for
  Replace $T_i$ by it's divided triangles.
  $k = k + 1$.
end while
return T
```

Algorithm 4 boundary_adjustment(points); Boundary adjustment of the produced Reference set by α-hull method

```
$(\bar{points}) = MDS(points)$.
Get boundary by $(\bar{H}) = \alpha - hull((\bar{points}))$.
Build Linear Interpolation model on the boundary data set $\bar{H}$ and place $N_H$ boundary points uniformly at regular distance on it.
Use this new set of boundary points and original intermediate points to form the data set $new\_points$.
return $new\_points$
```

Algorithm 5 boundary_adjustment(points); Boundary adjustment of the produced Reference set by edge checking method

```
Build Tringulation on the set of points to produce $tri1$.
for each $i \in points$ do
  Computing the set of connecting nodes neighbor$_i$
end for
for each edge $e(i, j) \in tri1$ do
  if size(neighbor$_i$ $\cap$ neighbor$_j$)=1 then
    Enlist Edge e(i,j) as the border.
  end if
end for
The collection of all the edges form the boundary set $\bar{H}$ of $N_H$ boundary points.
Build Linear Interpolation model on the boundary data set $\bar{H}$ and place $N_H$ boundary points uniformly at regular distance on it.
Use this new set of boundary points and original intermediate points to form the data set $new\_points$.
return $new\_points$
```

6 Results

Now with this updation strategy we form the code of the net algorithm. The code is written in Matlab and combined with already available code of SMS-EMOA. For simulation purpose we have simulated the code over DTLZ1 and DTLZ2 for different number of function evaluations. Here we present the approximated Pareto front generated by Δ_p SMS-EMOA over DTLZ1 for 100,000 function evaluations for 7 Dimensional problem in Figure 10 and for 100,000 function evaluations in DTLZ2 for 12 Dimensional problem in Figure 11. From the figures presented we can observe that the Pareto front produced is more or less uniform in nature. The mean of the Averaged Hausdorff distance over 10 runs for 100,000 function evalutions for DTLZ1 is 0.0186 and for DTLZ2 is 0.0522.

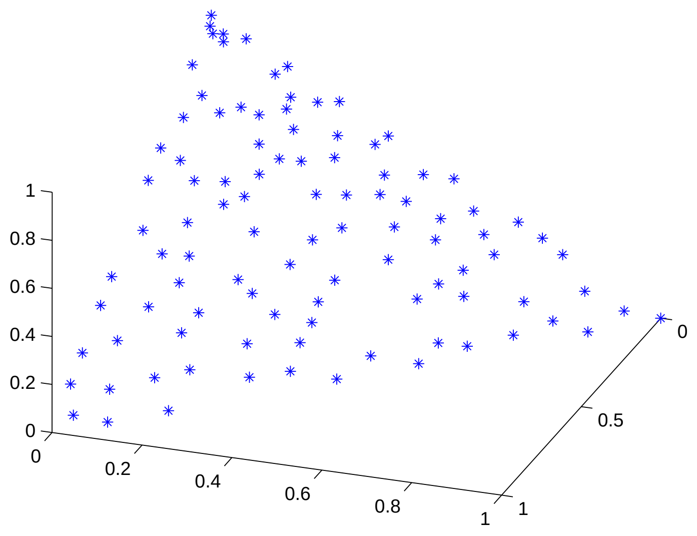

Figure 10: Approximated Pareto front for DTLZ1

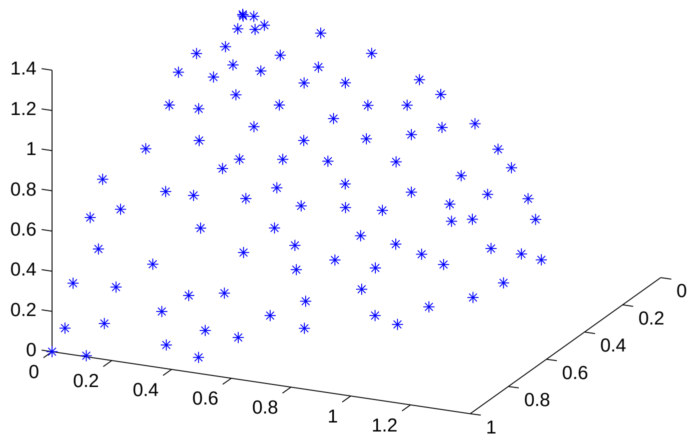

Figure 11: Approximated Pareto front for DTLZ2

7 Conclusion

The construction of equi-distributed reference set in 3-D was the main aim of this project work. Then producing appropriate updation strategy and combining the whole code was the next phase of the work. The results of construction of reference set is certainly satisfactory. Though the result of the net algorithm is fair, there may be other ways to improve the performance of the net algorithm with it. Basically the work was a more focus on computational geometry and producing a good updation strategy. In future there are lot of scope to work for the better performance of this algorithm. This whole project work was a nice experience for me and it helped me learn a lot in my way and gather a lot of knowledge.

References

[1] J.R. Berrendero, A. Cuevas, and B. Pateiro-López. A multivariate uniformity test for the case of unknown support. *Statistics and Computing*, 22(1):259–271, 2012.

[2] N. Beume, B. Naujoks, and M. Emmerich. Sms-emoa: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

[3] J.C. Bezdek, R. Ehrlich, et al. Fcm: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, 10(2-3):191–203, 1984.

[4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA)*, pages 825–830. Proceedings of the Congress on Evolutionary Computation (CEC-2002),(Honolulu, USA), 2002.

[5] V. Domiter and B. Žalik. Sweep-line algorithm for constrained delaunay triangulation. *International Journal of Geographical Information Science*, 22(4):449–462, 2008.

[6] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics, 2001.

[7] K. Gerstl, G. Rudolph, O. Schutze, and H. Trautmann. Finding evenly spaced fronts for multiobjective control via averaging hausdorff-measure. In *Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on*, pages 1–6. IEEE, 2011.

[8] J.A. Hartigan and M.A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[9] D.T. Lee and B.J. Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Parallel Programming*, 9(3):219–242, 1980.

[10] O. Schütze, X. Esquivel, A. Lara, and C.A.C. Coello. Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization. *Evolutionary Computation, IEEE Transactions on*, 16(4):504–522, 2012.