Using gradient-based optimization to create a hybrid genetic algorithm IIT Kharagpur

Harshal Dupare*, Amay Varma†,
Sophomores of the Department of Mathematics
IIT Kharagpur

I. Abstract

Being inspired by the Particle Swarm Optimization ([2] ,[3])and NSGA-II [5], we have explored the idea of a **hybrid algorithm for multi-objective optimization problem** trying to combine the benefits of both gradient-based and search-based approaches. Doing so we have revisited the idea of mutation and crossover in genetic algorithms and tried to incorporate the idea of local search and global search in an independent way. For **mutation** we generate a child solution by outputting a n-dimensional vector that is at some fixed distance from the parent. **Crossover** is managed by finding points of intersection between two hyperspheres, **Selection** is done by algorithms similar to ones that try to solve the vertex-cover problem approximately in polynomial time[11] and sorting on the basis of the number of objectives a solution dominates another. We then apply this hybrid algorithm on several classical problems presented in [1] and [4] and analyze the results.

II. Introduction

A. The problem of multi-objective optimization

There are mainly two kinds of algorithms used for multi-objective optimization. The first kind of algorithms are gradient-based, and the second kind is search-based. The principal difference between both algorithms is that gradient-based algorithms give very high preference to optimize in discrete steps towards the direction of the gradient or slope of the objective function. This optimization works best well for single-objective function optimization with one extremum and no other local extrema.

Search-based algorithms, on the other hand, give preference to better solutions using a 'fitness' parameter but do not, in most cases, optimize in the direction of gradient so as to establish diversity; this helps in searching for global extrema objective functions with multiple local extrema.

Genetic algorithms are nature-inspired search based algorithms that deal with the optimization of one or more objective functions given a fixed search space, they incorporate methods akin to biological evolution like genetic mutation, chromosomal crossover, and natural selection.

B. Comparison between gradient-based approach and search-based approach

We can infer from the above explanations and [10] that gradient-based algorithms are computationally less expensive on average but often falter when there are multiple local extrema and single global extrema or numerous objective functions.

The issue arises because when we give high preference to optimization in the direction of the gradient of the objective function, it is likely that we will converge around various local extrema instead of global extrema.

Searched-based algorithms like genetic algorithms, however, generally do not optimize towards the gradient. The question then arises, how do genetic algorithms narrow down the search space to find the global optimum? By establishing a 'fitness' parameter for a given parent solution, they tend to give some bias towards fitter solutions compared to less fit solutions, resulting in slower narrowing down of the search space; this works better in the case of multi-objective optimization and also in cases where there are multiple local extrema. This diversity comes at the cost of more computation and higher time complexity of the optimization algorithm.

C. Inspiration for the hybrid genetic algorithm

Particle Swarm Optimization [2] is a unique genetic algorithm that uses the 'velocity' of the population space, which is a notion of gradient. We use the word 'unique' as it is rare for genetic algorithms to use vector differentials instead of a fitness parameter to generate child solutions from a parent solution and the word 'notion' as the vector along which the child is created is not precisely the gradient of the objective function; instead, it is referred to as 'velocity' of the particle.

More formally, the velocity of the particle X_i in dimension d can be quantified as:

$V_{(i,d)} = \omega V_{(i,d)} + \phi_p R_p(p_{i,d} - x_{i,d}) + \phi_g R_g(g_d - x_{i,d})$

R_p and R_g are uniformly distributed random variables in the region $(0, 1)$

ω, ϕ_p, ϕ_g are hyperparameters.

$p_{i,d}$ denotes the best position found so far for particle x_i

g_d denotes the best position found so far for the entire swarm.

As we observe here, the concept of optimizing the parent solution by moving in a particular direction is similar to gradient-

based algorithms; one can now explore the possibilities of a hybrid algorithm that has the advantage of faster convergence which gradient-based algorithms provide along with having the diversity to deal with multi-objective functions with multiple local extrema.

D. Local search and global search

We first describe the concepts of *local search* and *global search* in a search-based algorithm before introducing our algorithm in the next section.

As observed in [8] Every search-based algorithm must have two search subroutines for it to work efficiently. Given a search space, the first subroutine is one that searches around a smaller sub-neighborhood of the parent to find the optimum child; this usually is a fast subroutine in most implementations. We name this subroutine 'local search'.

The second subroutine searches a region further away from local sub-neighborhood of the current generation's points; this is key to ensuring that the convergence isn't centric around local extrema and instead on the global extrema. Most genetic algorithms cover both local search and global search via mutation and crossover however there is a certain amount of overlap between the mutation and crossover process of many genetic algorithms, i.e. the local search can penetrate the global search and vice versa, resulting in computational inefficiency due to overlap of searching.

III. The hybrid algorithm

Like all genetic algorithms, our algorithm superficially follows a four step procedure.

1) Generate a random population
2) Perform mutation and crossover on random population to generate child solutions of parent population
3) From the combination of parent and child solutions, pick a subset to call the next generation
4) Repeat from (2) onwards on the next generation till convergence

What differentiates genetic algorithms from one another however is the various algorithms used for the *mutation* , *crossover* and *selection*subroutine.

Elaborating step by step, the first step is that of random population generation. A single member of the population is represented by 4 vector parameters.

Formally, if X is a member of the population: it is a 4-tuple of vectors, i.e.

$$X = (X_d, G_d, G_o, F_x)$$

X_d	Input or decision variable vector
G_d	Unit gradient vector of X_d
F_x	Output or objective variable vector
G_o	Gradient vector of F_x using G_d

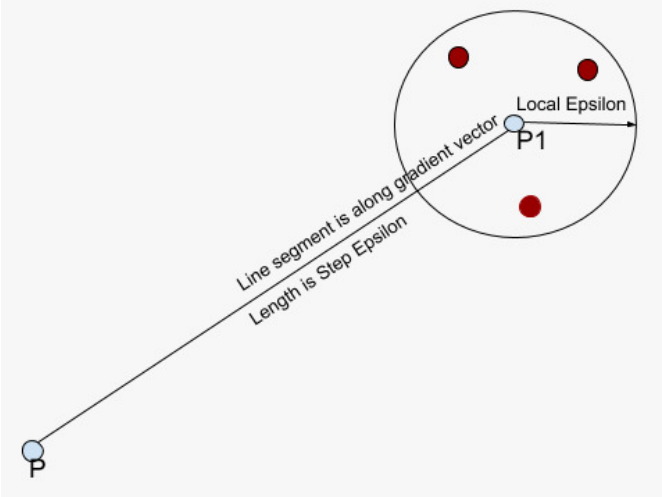
The algorithm ensures that $|G_d| = 1$, by vector calculus the gradient G_o can then be evaluated as:

$$G_o = \frac{F(X_d + \epsilon * G_d) - F(X_d)}{\epsilon}$$

ϵ is a hyperparameter that is given as input from the user.

A. Mutation

From parent **P**, we first travel a fixed distance **step epsilon** in the direction of the gradient and label this point **P1**. We now generate a n-dimensional hyper-sphere with radius **local epsilon** and centre **P1** and pick a random point on the inside of this sphere as child **C** Without loss of generality, consider a two-dimensional setup in the below diagram, all points shaded red are potential candidates for child **C**.



B. Crossover

Notation:

$\vec{C_1}$	Center of the first hypersphere
$\vec{C_2}$	Center of the second hypersphere
$R1$	Radius of the first hypersphere
$R2$	Radius of the second hypersphere
d	The distance between two centers
$\vec{X}$	Center of intersection of two hyperspheres

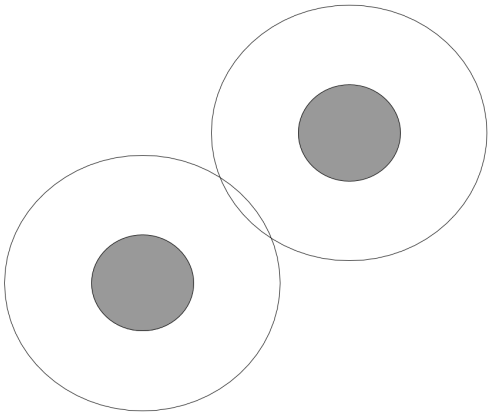
To understand the crossover algorithm clearly, it is imperative to first understand the reasoning behind why it was picked. We must ensure that our crossover algorithm finds children in a region where mutation cannot, thus securing an effective global search.

1) Understanding region of mutation: It is clear that from parent **P**, the furthest a child of mutation can be is Q = **local epsilon + step epsilon**.

Let us pick some parameter **minD = minimum distance** which is greater than Q.

Consider two parents P1 and P2, without loss of generality, consider a two-dimensional setup as seen in the below diagram. The two shaded circles are of radius **minD**. If we generate the child of crossover in any region outside the two shaded circles, it is ensured that **the child of crossover will never overlap with the child of mutation** which was our initial goal. This will make the crossover exclusively perform **global search** and

mutation perform **local search** resulting in greater efficiency.



2) Establishing the crossover algorithm: Let us consider some radii $R1$ and $R2$ which are the radii of the two hyperspheres.

Lemma: The region of overlap between two n-dimensional hyperspheres is a hypersphere of dimension $n-1$.

Proof: Translate and rotate the line joining the centers of the two hyperspheres so that it lies on the axis. Then, equations of hyperspheres:

$$\sum_{i=1}^{n} x_i^2 = R_1^2 \tag{1}$$

$$(x_1 - d)^2 + \sum_{i=2}^{n} x_i^2 = R_2^2 \tag{2}$$

Subtracting (2) from (1) we get the single point of intersection of the hyperspheres $x_1 = \frac{R1^2 - R2^2 + d^2}{2d}$

Plugging x_1 back in (1) we get the required hyper-curve of intersection which has the same form as that of a hyper-sphere of dimension $n-1$

QED.

However R_1 and R_2 cannot be arbitratrily picked. It can inferred from the above discussion that the two hyper-spheres must:

- Intersect with each other but one should not be completely inside another.
- The child of crossover of intersection must be atleast **minD** away from the two parents.

By applying basic conditions of geometry, one can find that:

- $R1 - R2 < d < R1 + R2$
- $\min(R1, R2) >=$ **minD**

We now have to evaluate the center of the hypersphere, [9] $\vec{X}$ of dimension $n-1$ which is the locus of overlapping points of the two n-dimensional hyperspheres. $\vec{X}$ will be our child of crossover of two parents.

Mathematical derivation of point of intersection:

$R1 = ||\vec{X} - \vec{C_1}||$ (1)
$R2 = ||\vec{X} - \vec{C_2}||$ (2)
(1) can be written equivalently as:

$(\vec{X} - \vec{C1}) \cdot (\vec{X} - \vec{C1}) = R1^2$
$\implies \vec{X} \cdot \vec{X} - 2\vec{C1} \cdot \vec{X} + \vec{C1} \cdot \vec{C1} = R1^2$ (3)
Similarly, from (2)
$\implies \vec{X} \cdot \vec{X} - 2\vec{C2} \cdot \vec{X} + \vec{C2} \cdot \vec{C2} = R2^2$ (4)
Subtracting (3) from (4)

$\implies 2 * \vec{X} \cdot (\vec{C1} - \vec{C2}) + \vec{C2} \cdot \vec{C2} - \vec{C1} \cdot \vec{C1} = R2^2 - R1^2$
Compare with equation of a hyperplane:
$\vec{N} \cdot \vec{X} = K$
We find that:
$\vec{N} = 2(\vec{C1} - \vec{C2})$
$K = R2^2 - R1^2 - \vec{C2} \cdot \vec{C2} + \vec{C1} \cdot \vec{C1}$
Hence we have found the equation of the hyperplane.

The equation of the common normal between the two spheres is:
$\vec{X} = \vec{C1} + \lambda(\vec{C2} - \vec{C1})$
$\lambda \in (0, 1)$ The point of intersection between the common normal and the hyper-plane will give the center of the hyper-sphere of intersection.

Solving, we get:

$$\lambda = \frac{K - \vec{C1} \cdot \vec{N}}{\vec{C1} \cdot (\vec{C2} - \vec{C1})}$$

By translating and rotating the common normal, one can find the radius of intersection of hypersphere R_i as

$$R_i = \sqrt{R1^2 - (\frac{R2^2 - R1^2 - d^2}{2d})^2}$$

Now that we have in terms of known parameters,

- Equation of hyperplane
- Center of hypersphere of intersection
- Radius of hypersphere of intersection

We can find the crossover of $\vec{P1}$ and $\vec{P2}$.

C. Selection

Establishing some hyperparameters which denote how many children of crossover and mutation will be generated and how many members of the population will be picked for the next generation, we can lay out the selection algorithm as follows: Assume a K dimensional space, Give preference to a parent $\vec{P}$ over parent $\vec{Q}$ to go to the next generation if $\vec{P}$ dominates $\vec{Q}$ in more than $\frac{K}{2}$ dimensions. To ensure that we do not have solutions that are clustered around a region (as this does not help in increasing diversity) we also remove solutions that are closer than some fixed minimum distance. To do this, we first view this data as a graph with vertices representing points and edges having weight equivalent to the euclidean distance between points. Now all distances that are less than some fixed minimum distance should be removed, more formally: We must remove a subset of nodes such that all edges are covered, which reduces to the vertex cover problem. Since the vertex cover problem is NP-hard, we used efficient approximations as in [11] to solve it in polynomial time.

IV. Results

We now apply this algorithm on classical problems used in [x] and [y] and analyze the results.

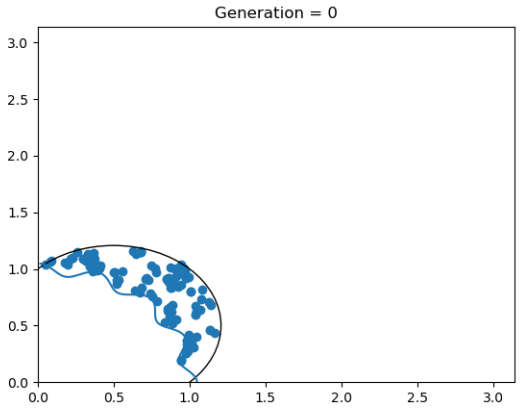
A. TNK problem

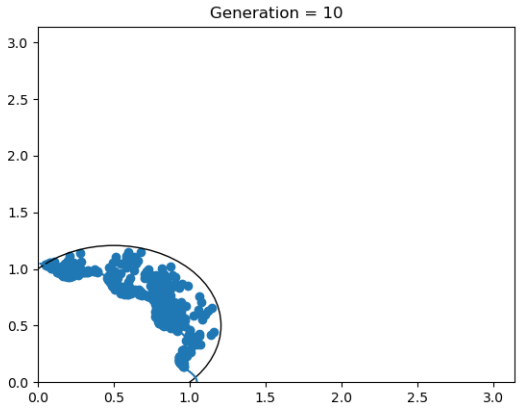
This is a two objective optimization problem with two input variables, however there are non-linear constraints set on the decision space variables, formally the problem can be described as:

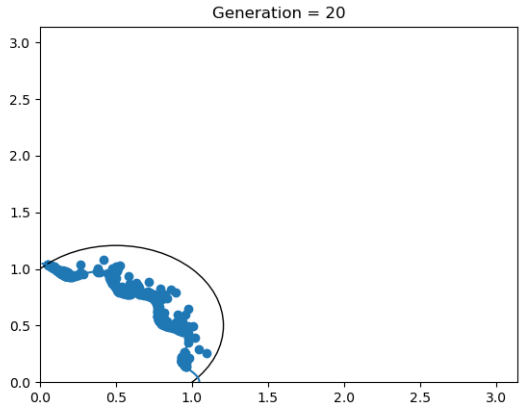
$$f_1 = x_1 \tag{3}$$

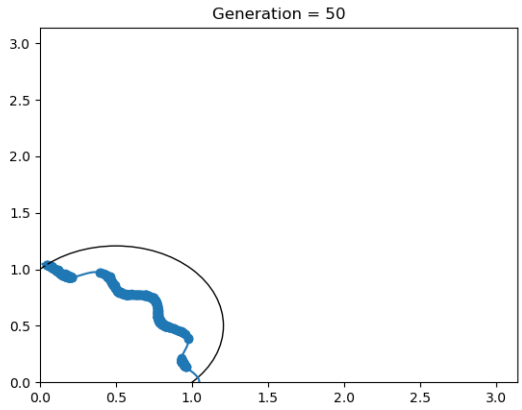
$$f_2 = x_2 \tag{4}$$

where
$x_1 \in [0, \pi], x_2 \in [0, \pi]$
Given the constraints:
$g_1 = -x^2 - x_2^2 + 1 + 0.1\cos\left(16 \arctan \frac{x_2}{x_1}\right) \leq 0$
$g_2 = (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5$









The analysis for the DEB and TNK problem will be done jointly in a later subsection.

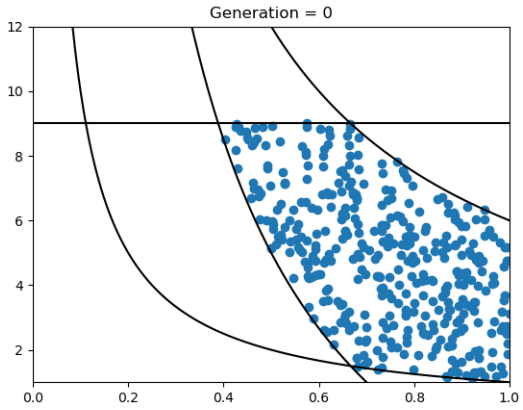
B. DEB problem

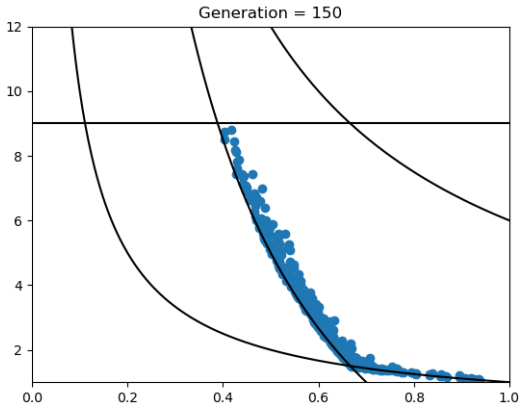
Similar to the TNK problem this is also a two objective optimization problem with two input variables with non-linear constraints set on the decision space variables, formally the problem can be described as:

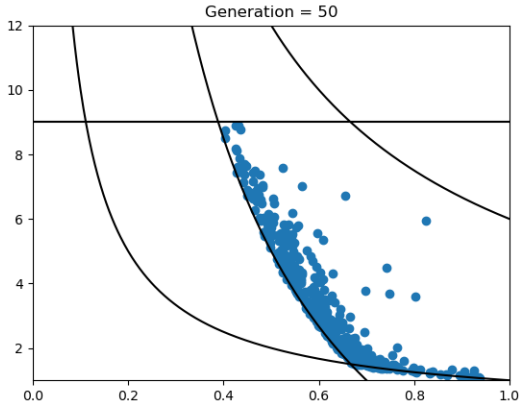
$$f_1 = x_1 \tag{5}$$

$$f_2 = \frac{1 + x_2}{x_1} \tag{6}$$

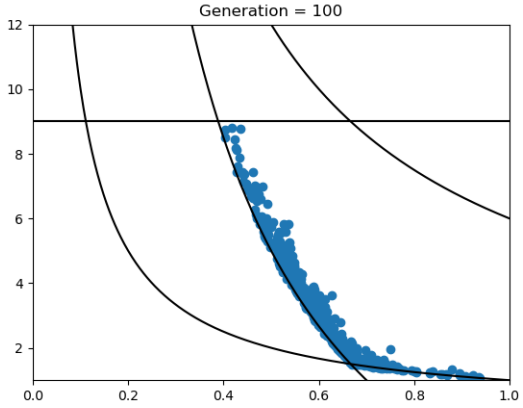
where
$x_1 \in [0.1, 1], x_2 \in [0, 5]$
Given the constraints:
$g_1 = x_2 + 9x_1 \geq 6$
$g_2 = -x_2 + 9x_1 \geq 1$



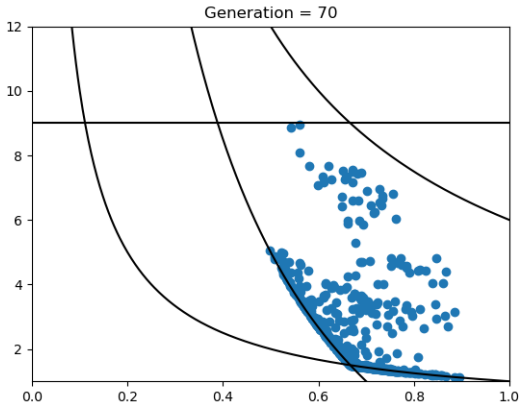




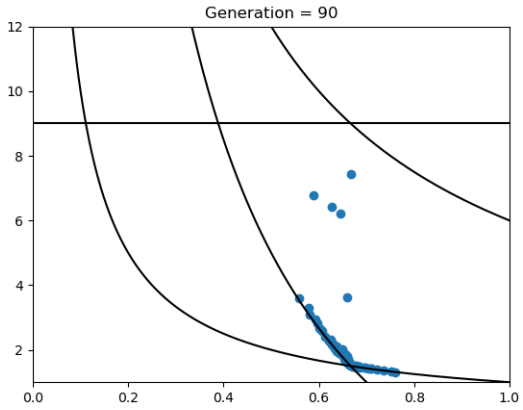
C. Joint analysis of both problems with some failed initial attempts and further exploration



A better result would be one where the solutions (denoted by blue points) cover the entire pareto front (showing diversity). We observed that as hypothesized, when the crossover parameters were made overlapping with the mutation parameters, due to failure of global search, solutions around local optima were being preferred and the algorithm failed to converge around the pareto front.



Another failure of global search was when **local epsilon** was considerably smaller than **step epsilon** which resulted in lack of diversity and hyper convergence of solutions around a small neighbourhood:



As expected, the best results were found when **local epsilon** dominated **step epsilon** in the early generations, and in later generations the gradient-based **step epsilon** increased greater than **local epsilon**. This resulted in the fast convergence of the gradient-based approach and diversity provided by search-based genetic algorithm approach. It is also important to note that when **minD** was greater than **local epsilon + step epsilon** is when the convergence was highly accelerated due to zero overlap between crossover and mutation as previously hypothesized.

V. Conclusion

We decided to explore the possibility after being inspired by the Particle Swarm Optimization [2] to evolve the differential evolution concept even further by using concepts behind popular gradient based approaches [6] and [7] to create a hybrid genetic algorithm. Along with the idea of hybridness, in most present genetic algorithms we studied it was observed that children of crossover and mutation sometimes overlapped because the feasible region of a child of crossover had overlap between the feasible region of a child of mutation. Hence with this algorithm we managed to succesfully explore:

- Creating a crossover and mutation algorithm where the child of crossover is always distinct from the child of mutation, in this case mutation intends to perform the local search while crossover intends to perform global search.
- A hybrid search-based genetic algorithm incorporating properties of gradient-based algorithms to make the algorithm have the speed of gradient-based approaches and the diversity of search-based approaches.

Perhaps in the future, there will be more hybrid algorithms and greater overlap between researchers of gradient-based algorithms and search-based algorithms.

VI. References

[1] Rigoni, Enrico Poles, Silvia. (2005). NBI and MOGA-II, two complementary algorithms for Multi-Objective optimizations.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942-1948 vol.4, doi: 10.1109/ICNN.1995.488968.

[3] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, USA, 1998, pp. 69-73, doi: 10.1109/ICEC.1998.699146.

[4] Rigoni, E. 2004, Hole functions problem, ESTECO Technical Report 2004-002, http://www.esteco.com.

[5] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," in IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182-197, April 2002, doi: 10.1109/4235.996017.

[6] Liu, Xin Reynolds, Albert. (2015). Gradient-based multi-objective optimization with applications to waterflooding optimization. Computational Geosciences. 20. 10.1007/s10596-015-9523-6.

[7] Peitz Dellnitz. (2015). Gradient-Based Multiobjective Optimization with Uncertainties https://arxiv.org/pdf/1612.03815.pdf

[8] Weise, T., Wu, Y., Chiong, R. et al. Global versus local search: the impact of population sizes on evolutionary algorithm performance. J Glob Optim 66, 511–534 (2016).

[9] Michel Petitjean. Spheres Unions and Intersections and Some of their Applications in Molecular Mod-eling. Antonio Mucherino, Carlile Lavor, Leo Liberti, Nelson Maculan. Distance Geometry: Theory,Methods, and Applications., Springer New York, pp.61-83, 2013, 978-1-4614-5127-3. 10.1007/978-1-4614-5128-0_4.$hal - 01955983$

[10] F. Ahmad, N. A. M. Isa, M. K. Osman and Z. Hussain, "Performance comparison of gradient descent and Genetic Algorithm based Artificial Neural Networks training," 2010 10th International Conference on Intelligent Systems Design and Applications, Cairo, 2010, pp. 604-609, doi: 10.1109/ISDA.2010.5687199.

[11] Khamayseh, Yaser Mardini, Wail Shatnawi, Anas. (2012). An Approximation Algorithm for Vertex Cover Problem

Acknowledgment

The authors would like to deeply thank Dr. Nirupam Chakraborti our professor for taking this course and motivating us to dutifully attend our classes and grasp relevant material along with explaining the concepts in a lucid manner and encouraging interaction. Without his mentorship it would be impossible to formulate the ideas presented in this paper. We would also like to thank the teaching assistants who helped run the course smoothly, Mr. Swagata Roy and Mr. Bashista Kumar Mahanta.