Parallel Semi-supervised Multi-objective Optimization using Convex Hulls in a low dimensional decision space

Ishrak Hayet Email: ihayet@ncsu.edu

I. Introduction

In this study, we try to use the SWAY optimizer [1] as a baseline on 11 different datasets. Then we propose a new optimization method based on dimensionality reduction and topological data analysis using convex hull. The proposed method eventually uses a similar domination based optimizer like SWAY. The method uses Zitzler's domination predicate [cite] to select the best cluster. Eventually, we use a discretization based explainer to find interesting good and bad clusters from the results of the optimizer.

In the SWAY paper [1], the authors focus on using a random sampling approach coupled with dominance determination to effectively combine the notion of clustering and optimization. The core tenet of this methodology is that it is easy and affordable to look at many different values in the decision space. However, it is difficult and expensive to look at many values in the objective space. Especially, in the case of search based software engineering (SBSE), looking at all sample and combinations of features to find an optimal set of objectives is not feasible [2]. For example, the GNU Compilation Collection (GCC) has numerous optimization options for its compiler [3]. The combination of the different optimization flags of GNU (GCC) can easily exceed the thousands [3]. So, the feasible alternative to explore all option combinations, is to sample a few possible combinations and analyze the compiler's performance based on those combinations. And at the same time, the goal would be to find an optimal cluster that those combinations belong to so that we can eventually generalize optimal combinations of flags using some form of explanation. The authors in the SWAY paper [1] try to utilize a combined clustering and optimization mechanism to keep dividing the data and at each step select the division that optimizes a certain set of objectives. The authors utilize a method called FastMap [cite] to perform clustering of the samples with respect to the decision space and eventually utilize Zitler's predicate of domination [cite] to sway into one of the clusters that best optimizes the objectives. This method is performed recursively until the cluster size becomes equal to a set number.

FastMap clustering combined with SWAY optimizer performs very well as a baseline optimizer because it is very fast and produces optimal clusters that are on par with and in some cases better than those produced by complex time consuming methods [1]. However, one critical drawback of SWAY can surface for very large datasets where we have to sequentially cluster and sway many times. Since the recursive next step of FastMap clustering depends on what we get from the current FastMap step and SWAY optimization, the combination of FastMap and SWAY optimization becomes sequential and can potentially become slow on very large datasets. One might argue that SWAY converts the optimization problem into a semi-supervised multi-objective optimization where we are not required to know the complete objective space but rather we only randomly choose an initial pool of samples and perform the optimization on that much smaller pool. Even with seed variation and running SWAY multiple times each time with a different set of initial pool, the results seem to be unstable sometimes. Increasing the sample size stabilizes the outcomes. However, with an increased sample size, the sequential nature of recursive clustering and optimization adopted by FastMap and SWAY becomes a critical bottleneck.

So, in this paper, we propose an slightly different method of clustering and optimization that does not rely on recursive operation but instead tries to divide the decision space into multiple quadrilateral and overlapping sub-spaces. After asynchronously ensuring local optimality using Zitzler predicate [4] in each sub-space, the locally optimal clusters are compared amongst themselves again using Zitzler's predicate [4] and eventually a globally optimal cluster emerges. The idea was motivated by topological data analysis where the tenet is to derive global insights from local information [13]. In our method, we begin with the actual shape of the data in a lower dimensional decision space since it is widely established that the shape of data has meaning and can help in analyzing the data [15][14]. This method of divide-and-conquer helps the process of combined clustering and optimization to be run in parallel and then to be eventually merged. Moreover, we wish to gain more control over the randomness that is associate with SWAY [1]. Therefore, we are motivated to exploit the shape of the data using its convex hull to determine the vertices of the overlapping quadrilateral sub-spaces. Now, computing convex hull of high dimensional data is quite time-consuming. And, according to [5] and [**?**], for solutions in high-dimensional space, the objective space is sparsely distributed. As a result, clustering and optimizing the objective space becomes challenging in high-dimensional space. Because of these two issues of convex hull computational time and sparse objective space, we utilize dimensionality reduction method Principal Component Analysis (PCA) to reduce the data dimensionality to two. For determining the locally optimal cluster within each quadrilateral sub-space, we consider only the points that are located within a specific quadrilateral sub-space. Then we draw four lines each of which goes through the centroid of the quadrilateral and the center of each side. We project all the bounded points on each of these four lines using cosine projection. We perform this projection for all the quadrilaterals. Then, we use Zitzler's predicate [4] to find the best projection from all the projections within all the quadrilaterals.

II. Related Work

In [2], the authors propose a method for exploring compiler optimization sequences using clustering-based selection. Compiler optimizations are crucial for improving the performance of software applications. However, choosing the best optimization sequence for a specific program can be challenging, as the search space is typically vast and complex [2]. The authors introduce a clustering-based selection technique that aims to reduce the exploration time and effort by grouping similar programs together. By leveraging these groupings, the proposed method can effectively explore and select the most promising optimization sequences. This approach should lead to improved performance and efficiency when compared to traditional, exhaustive search methods or heuristics [2].

In [7], the authors focuse on surrogate-based optimization (SBO) techniques for solving expensive multimodal optimization problems. Multimodal problems involve multiple optimal solutions, which makes them challenging to solve, especially when the objective functions are computationally expensive to evaluate. The authors propose a novel approach that combines surrogate-based optimization with clustering-based space exploration [7]. This method aims to improve the efficiency of the optimization process by leveraging the benefits of both techniques. Surrogate-based optimization relies on surrogate models (e.g., Gaussian process regression, kriging) to approximate the expensive objective functions, thus reducing the number of expensive evaluations required. Clustering-based space exploration helps in identifying and exploring multiple promising regions of the search space, which is crucial for multimodal problems. The proposed method involves building surrogate models for the objective functions, using clustering algorithms to partition the search space into sub-regions, and then applying SBO techniques within each sub-region. This approach enables the algorithm to explore multiple optima simultaneously while reducing the number of expensive function evaluations required [7].

Inspired by SWAY [1], we have used random cosine projections in the proposed method. Random projection is a mathematical technique used in machine learning, data mining, and signal processing to reduce the dimensionality of high-dimensional data while preserving certain properties of the data [8]. The basic idea is to randomly project the high-dimensional data onto a lower-dimensional subspace, while preserving the pairwise distances between the data points as much as possible. Random projection has many applications in various fields, including computer vision, natural language processing, and bioinformatics. It is a simple and efficient way to reduce the computational cost and storage requirements of algorithms that operate on high-dimensional data. There are several methods for performing random projection, including Gaussian random projection and sparse random projection. These methods differ in the way that random projections are generated, and they have different properties and applications [8].

In the proposed method, we have considered some heuristics like using boundary points instead of random starting points, using centroid to guide the projection line, using overlapping quadrilateral sub-spaces instead of any other polygon etc. Utilizing heuristics is an established tool in research [9]. In the paper [9], the authors argue that heuristics can be effective because they are adapted to the structure of the environment in which they are used. They suggest that people often rely on heuristics because they are well-suited to the statistical properties of the environment and the cognitive limitations of human decision-makers. The paper provides examples of heuristics that are commonly used by people, such as the recognition heuristic, which involves making decisions based on familiarity or recognition, and the take-the-best heuristic, which involves using a simple rule to choose between two options based on the first available piece of information. The authors also discuss the results of several studies that support the effectiveness of heuristics. They argue that heuristics can lead to high levels of accuracy, especially when used in environments with certain statistical regularities [9].

In the paper [14] published in Nature Scientific Reports, the authors explore the potential of using topology, a branch of mathematics that deals with properties of spaces that are preserved under continuous deformations, as a tool for extracting insights from complex and high-dimensional data sets. The author presents an overview of the emerging field of topological data analysis (TDA) and discusses its potential applications and benefits. The authors argue that topology can offer a powerful framework for understanding the shape and structure of such data, by focusing on the features that are invariant under continuous deformations [14]. The paper emphasizes the value of topology as a tool for data analysis, offering a new perspective on the shape and structure of complex data [14]. Inspired by how topological data analysis is based on analyzing the shapes of data, we utilize Principal Component Analysis to convert the data into a 2-dimensional data and then use Convex Hull algorithm Quickhull to find the shape of data [16]. Principal Component Analysis (PCA) is a widely used statistical technique for dimensionality reduction, data compression, and data visualization. It is a linear transformation method that aims to identify and project the data onto a lower-dimensional subspace while preserving the maximum amount of variance in the original data [10]. PCA is particularly useful when dealing with high-dimensional data, where it helps to simplify the data, reduce computational complexity, and identify underlying patterns or structures [10]. In the paper [11], the authors have shown that performing dimensionality reduction before clustering improves the performance of clustering. A convex hull is a geometric concept that can be defined as the smallest convex set or polygon that contains all the points of a given set of points in a Euclidean space [12]. In other words, it is the tightest convex shape that encloses all the points in the given set. The convex hull can be thought of as the shape that would be formed by stretching a rubber band around the outermost points of the set, so that it snaps tightly around them. The convex hull is an important concept in computational geometry and has applications in various fields, such as computer graphics, collision detection, pattern recognition, and optimization problems. There are several algorithms available for computing the convex hull of a set of points, such as the Graham scan, Jarvis march, and QuickHull algorithms [12].

III. Methodology

A. Preprocessing

We have preprocessed the data to remove any sample that has a missing value in any one feature. Also, we performed normalization on the dataset to account for scaling discrepancies among different features.

B. Principal Component Analysis

Principal Components Analysis (PCA) is a linear transformation technique used for reducing the dimensionality of a dataset while maintaining as much of its original variance as possible. In this report, we demonstrate the application of PCA for reducing the dimensionality of a dataset to two principal components. This allows us to visualize the data in a two-dimensional space, providing insights into the underlying structure and relationships between variables. PCA aims to find a new coordinate system, represented by orthogonal axes called principal components, that maximizes the variance captured by the projected data points. The first principal component (PC1) accounts for the most variance, followed by the second principal component (PC2), and so on.

Given a dataset X with n observations and p variables, the steps to perform PCA and reduce the data to two principal components are as follows:

1) Standardization: First, standardize the dataset by subtracting the mean and dividing by the standard deviation for each variable. This ensures that each variable contributes equally to the analysis.

$$X_{std} = (X - \mu)/\sigma \tag{1}$$

Where μ is the mean and σ is the standard deviation.

2) Covariance Matrix Calculation: Compute the covariance matrix (C) of the standardized data (X_{std}). The covariance matrix is a symmetric, square matrix that measures the linear dependence between pairs of variables.

$$C = (1/(n-1)) * X_{std}^T * X_{std} \tag{2}$$

3) Eigenvalue Decomposition: Perform eigenvalue decomposition on the covariance matrix C to obtain the eigenvectors and their corresponding eigenvalues. The eigenvectors represent the direction of the principal components, while the eigenvalues represent the variance explained by each principal component.

$$\lambda, V = eig(C) \tag{3}$$

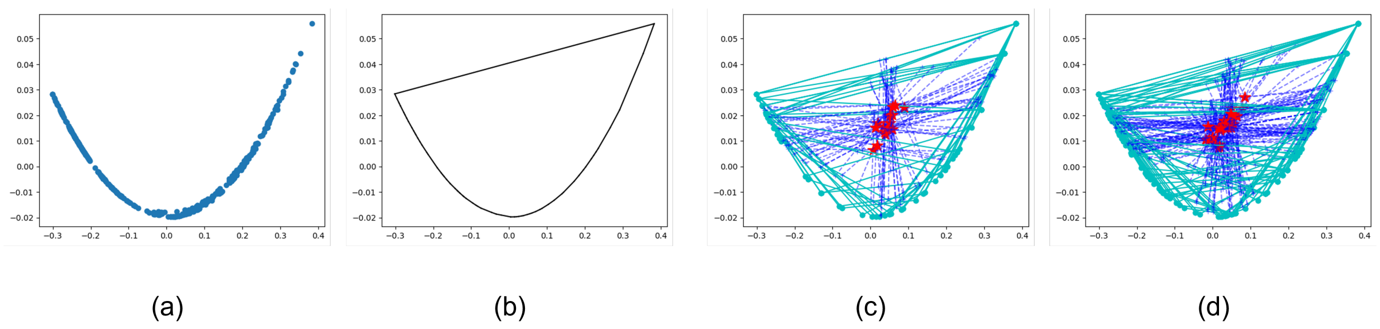

Fig. 1. (a) auto93.csv after PCA (b) Convex Hull of auto93.csv in 2d Principal Component Space (c) Sampling $|ConvexHullVertices|/4$ points from each quadrant (d) Sampling $2 * |ConvexHullVertices|/4$ points from each quadrant

4) Select Two Principal Components: Sort the eigenvalues in descending order and choose the top two eigenvectors associated with the two largest eigenvalues. These eigenvectors represent the first and second principal components (PC1 and PC2).

$$PC1 = V[:,0]PC2 = V[:,1] \tag{4}$$

5) Projection Onto Principal Components: Project the standardized data (X_std) onto the first two principal components (PC1 and PC2) to obtain the reduced data.

$$Y = X_{std} * [PC1, PC2] \tag{5}$$

The reduced dataset (Y) now contains two principal components, which explain the highest amount of variance in the original dataset. By plotting the data points in the two-dimensional space spanned by PC1 and PC2, we can visualize patterns and relationships within the data that may not have been apparent in the original high-dimensional space.

C. Finding Boundary Points

The convex hull of a set of points is the smallest convex polygon that contains all the points. It represents the boundary of the dataset and is essential in various applications, such as computational geometry, pattern recognition, and computer vision. The QuickHull algorithm is an efficient method for computing the convex hull of a 2D dataset. In this section, we demonstrate the application of the QuickHull algorithm to a 2D dataset, focusing on the detection of the convex hull and the identification of its boundary points.

1) QuickHull Algorithm: The QuickHull algorithm is a divide-and-conquer approach that recursively divides the dataset into subsets based on their relative positions to the current convex hull. The algorithm starts by finding the leftmost (minimum x) and rightmost (maximum x) points, which are guaranteed to be part of the convex hull. Then, it recursively processes the subsets of points above and below the line formed by these two points, finding the furthest point from the line in each subset and adding it to the convex hull.

The key steps of the QuickHull algorithm are as follows:

- Find the leftmost and rightmost points, A and B, in the dataset.
- Divide the dataset into two subsets, S1 and S2, where S1 contains the points above the line AB, and S2 contains the points below the line AB.
- Recursively find the convex hull for each subset.

The recursive function of the QuickHull algorithm takes a set of points S and two points, P and Q, as inputs. The function finds the convex hull of the points in S that lie outside the triangle formed by points P and Q.

- If S is empty, return an empty hull.
- Find the point C in S that is farthest from the line segment PQ.
- Remove C from S.
- Divide S into two subsets, S_{left} and S_{right}, where S_{left} contains the points to the left of the line segment PC, and S_{right} contains the points to the right of the line segment CQ.
- Recursively compute the convex hull for S_{left} and S_{right} with points P, C, and Q.
- Combine the results to form the final convex hull.

2) Distance Calculation: To find the point farthest from a line segment, we use the following formula to calculate the distance d between a point $M(x_m, y_m)$ and a line segment PQ with points $P(x_p, y_p)$ and $Q(x_q, y_q)$.:

$$d(M, PQ) = \frac{|(y_q - y_p)x_m - (x_q - x_p)y_m + x_q y_p - y_q x_p|}{\sqrt{(y_q - y_p)^2 + (x_q - x_p)^2}} \tag{6}$$

D. Forming Overlapping Quadrilateral Sub-spaces

Using the convex hull vertices, we form multiple overlapping quadrilateral sub-spaces using the following process.

1) Sampling Boundary Points: We divide the 2D space into quadrants and sample equal number of points from the sections of the convex hull that lie within each quadrant. This approach allows for the representation and analysis of spatial data with a balanced distribution of points across the quadrants, ensuring that each region is equally represented. Since we normalize the data between $[-1, 1]$, the center of the four quadrants is $(0, 0)$. Then we identify and divide the vertices of the convex hull into four different collections of vertices depending on their position in the quadrants. Then, from each quadrant we randomly sample $2 * TotalVertices/4$ points. If a quadrant has $< 2 * TotalVertices/4$ number of vertices, then we randomly sample from that quadrant with replacement.

2) Projection: After randomly sampling 4 vertices from 4 quadrants, we form a quadrilateral using those 4 vertices and try to find the centroid of the quadrilateral using the four triangles method [17]. At first, we find the centroid of each of the four triangles formed by drawing the two diagonals of the quadrilateral. We use the following equation to compute the centroid of triangle i:

$$x_c^i, y_c^i = \frac{x_1 + x_2 + x_3}{3}, \frac{y_1 + y_2 + y_3}{3} \tag{7}$$

We use the following equation to find the intersection between the lines formed by opposing triangles:

$$C_x = \frac{(x_1 y_2 - y_1 x_2)(x_3 - x_4) - (x_1 - x_2)(x_3 y_4 - y_3 x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \tag{8}$$

TABLE I
COMPARING MEAN AND STD OF THE BEST CLUSTERS FOR ALL, SWAY1, AND SWAY2

	all	sway1 (best)	sway2 (best)
auto2	CityMPG+ 21 (4.68) Class- 18.2 (9.57) HighwayMPG+ 28 (5.07) Weight- 3050 (630.859)	CityMPG+ 39 (8.20) Class- 9.1 (0.97) HighwayMPG+ 43 (6.64) Weight- 2240 (205.078)	CityMPG+ 40 (7.82) Class- 8.5 (2.25) HighwayMPG+ 41 (3.52) Weight- 2437 (400.53)
auto93	Acc+ 15.5 (2.73) Lbs- 2807 (893.75) Mpg+ 20 (7.81)	Acc+ 15.5 (2.26) Lbs- 2678 (425.00) Mpg+ 30 (7.81)	Acc+ 15.1 (2.02) Lbs- 2524 (605.35) Mpg+ 40 (6.52)
china	N_effort- 2109 (3829.29)	N_effort- 241 (148.438)	N_effort- 1532 (3552.67)
coc1000	AEXP- 3 (1.17) EFFORT- 19641 (27666.01) LOC+ 1016 (632.42) PLEX- 3 (1.56) RISK- 5 (5.46)	AEXP- 2 (1.56) EFFORT- 43724 (37663.28) LOC+ 1363 (707.81) PLEX- 4 (2.23) RISK- 11 (6.64)	AEXP- 3 (1.85) EFFORT- 22052 (25453.66) LOC+ 1463 (853.14) PLEX- 3 (2.23) RISK- 6 (4.53)
coc10000	Effort- 21463 (28705.07) Loc+ 931 (628.90) Risk- 5 (5.46)	Effort- 16928 (28136.32) Loc+ 1035 (639.06) Risk- 6 (4.68)	Effort- 18523 (28301.53) Loc+ 953 (583.35) Risk- 5 (4.89)
health-easy	ACC+ -12.24 (5.06) MRE- 119.33 (47.39) PRED40+ 0 (32.55)	ACC+ 0 (0.73) MRE- 0 (40.03) PRED40+ 83.33 (32.55)	ACC+ -4.52 (2.87) MRE- 38.53 (45.29) PRED40+ 67.38 (27.79)
health-hard	ACC+ 7.16 (3.23) MRE- 75.4 (11.15) PRED40+ 25 (14.64)	ACC+ 7.73 (0.883) MRE- 73.14 (2.84) PRED40+ 25 (9.76)	ACC+ 5.14 (2.27) MRE- 78.51 (8.81) PRED40+ 25 (16.78)
nasa93dem	Defects- 2077 (3225.39) Effort- 278 (678.32) Kloc+ 48.5 (102.93) Months- 21.5 (10.312)	Defects- 1553 (1605.85) Effort- 210 (184.60) Kloc+ 38 (26.01) Months- 21.3 (5.85)	Defects- 2253 (2852.55) Effort- 269 (550.23) Kloc+ 42.37 (87.67) Months- 22.1 (8.39)
pom	Completion+ 0.9 (0.09) Cost- 313.834 (188.60) Idle- 0.247 (0.208)	Completion+ 0.899 (0.1) Cost- 215.11 (85.80) Idle- 0.182 (0.189)	Completion+ 0.85 (0.14) Cost- 267.21 (103.76) Idle- 0.25 (0.195)
SSM	NUMBERITERATIONS- 6 (8.59) TIMETOSOLUTION- 133.55 (109.96)	NUMBERITERATIONS- 5 (1.95) TIMETOSOLUTION- 134.86 (55.21)	NUMBERITERATIONS- 6 (10.02) TIMETOSOLUTION- 139.63 (105.43)
SSN	Energy- 1239.253 (1288.70) PSNR- 45.72 (5.55)	Energy- 1413.825 (594.74) PSNR- 45.925 (5.06)	Energy- 1330.352 (1288.70) PSNR- 45.51 (4.35)

$$C_y = \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \qquad (9)$$

Then, we draw four different lines between each of the side's centers and the quadrilateral's centroid. These four lines will eventually be used as projection lines for each quadrilateral. Then, for a quadrilateral, we identify all the points that are bounded by the quadrilateral by checking if the sum of areas of the four triangles formed using vertex pairs of the quadrilateral and a particular point is equal to the area of the quadrilateral. This method works only for convex polygons and since each quadrilateral is bounded within the convex hull this method can be applied to find the points bounded by a quadrilateral.

E. Optimization

We use the Zitzler predicate [4] to compare all the projection lines of all the quadrilaterals. We count the total number of samples for which one projection was better than the rest using Zitzler predicate [4]. Then, we select the line with the highest count of better samples. This process selects only a fraction of the total samples. Then from within this fraction, we calculate the root mean squared error with respect to the minimum or maximum values of each column depending on the objective of optimization. We utilize a sliding window of variable length to find the sequences that further optimize the objectives.

IV. EVALUATION

A. Dataset

We evaluated and compared our method with the original SWAY method [1] for 11 datasets. The datasets were auto2.csv, auto93.csv, china.csv, coc1000.csv, coc10000.csv, health-easy.csv, health-hard.csv, nasa93dem.csv, pom.csv, SSM.csv, SSN.csv. Except for coc10000.csv and SSM.csv, the rest of the datasets were relatively small and our method was as fast as SWAY [1]. However, for the relatively bigger datasets, our method was comparatively slow at the step of pairwise comparing Zitzler's predicate for the different projections.

B. Results

In the following discussion, an improvement of performance is referred to as the reduction of standard deviation and optimization of the objective means.

1) SWAY2: We are comparing the mean and the standard deviation of the objectives for each dataset and for both methods.

2) XPLN2: Decision trees were used to explore the dataset as an alternative to rule learning as in XPLN1. For all the datasets, the average entropy for the nodes was approximately 0.45 with varying heights of the decision tree.

3) February Study: The number of quadrilaterals formed by sampling equal number of points from the different quadrants was considered as the budget for the study. Increasing the budget translates to increasing the number of points that are sampled from each quadrant at the expense of oversampling. Initially, B_0 was set as the $|ConvexHullVertices|/4$. Because of poor performance, B_1 was set as $4 * B_0$ in January. But then in February, we found out $B_2 = 2 * B_0$ would perform almost as well as B_1. So, the final value of the number of points sampled from each quadrant is $2 * |ConvexHullVertices|/4$

4) Ablation Study: When we do not include the last component of using a sliding window to further optimize using root mean squared error, the performance would deteriorate.

5) HPO Study: Number of initial random samples, cliff's delta threshold, and number of quadrilaterals were considered as hyper-parameters. Higher number of initial random samples led to better performance. Lower cliff's delta threshold of 0.147 would yield more 'false' values between all best and best rest, so the effect size is negligible. The higher the number of quadrilaterals the better the performance was at the expense of longer computational time, finding the balance of which is discussed in the February study.

V. Conclusion

In this report, we present a new method of using convex hull based asynchronous analysis to cluster and optimize a dataset at the same time in a low dimensional space. For most of the datasets, the new method performs better than the average of all the samples. In many cases, the new method also performs better than SWAY [1]. **One critical threat to validity** is that the projection analysis was meant to be run asynchronously across multiple compute nodes to improve computational time. However, due to lack of resources and time, improvements of parallelization was not experimented. **Future directions** would be to improve the pairwise comparison time of the different projection lines. Also, a learning component can be added to the projection lines so that for each iteration based on the root mean squared error between the objective average and the objective optima, the slope of the projection lines can change to accommodate the loss function.

References

[1] Chen, Jianfeng, et al. ""Sampling" as a baseline optimizer for search-based software engineering." IEEE Transactions on Software Engineering 45.6 (2018): 597-614.

[2] Luiz G. A. Martins, Ricardo Nobre, João M. P. Cardoso, Alexandre C. B. Delbem, and Eduardo Marques. 2016. Clustering-Based Selection for the Exploration of Compiler Optimization Sequences. ACM Trans. Archit. Code Optim. 13, 1, Article 8 (April 2016), 28 pages. https://doi.org/10.1145/2883614

[3] https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html. Last accessed: 04.15.23

[4] Zitzler, Eckart, Marco Laumanns, and Stefan Bleuler. "A tutorial on evolutionary multiobjective optimization." Metaheuristics for multiobjective optimisation 535.535 (2004): 21-40.

[5] Bao, Chunteng, Lihong Xu, and Erik D. Goodman. "A new dominance-relation metric balancing convergence and diversity in multi-and many-objective optimization." Expert Systems with Applications 134 (2019): 14-27.

[6] Cheng R, Jin Y, Olhofer M, Sendhoff B. Test Problems for Large-Scale Multiobjective and Many-Objective Optimization. IEEE Trans Cybern. 2017 Dec;47(12):4108-4121. doi: 10.1109/TCYB.2016.2600577. Epub 2016 Aug 26. PMID: 28113614.

[7] Dong, H., Song, B., Wang, P. et al. Surrogate-based optimization with clustering-based space exploration for expensive multimodal problems. Struct Multidisc Optim 57, 1553–1577 (2018). https://doi.org/10.1007/s00158-017-1826-x

[8] https://en.wikipedia.org/wiki/Random_projection. Last accessed: 04.15.23

[9] Gigerenzer, Gerd. "Why heuristics work." Perspectives on psychological science 3.1 (2008): 20-29.

[10] https://en.wikipedia.org/wiki/Principal_component_analysis. Last accessed: 04.15.23

[11] Allaoui, Mebarka, Mohammed Lamine Kherfi, and Abdelhakim Cheriet. "Considerably improving clustering algorithms using UMAP dimensionality reduction technique: a comparative study." Image and Signal Processing: 9th International Conference, ICISP 2020, Marrakesh, Morocco, June 4–6, 2020, Proceedings 9. Springer International Publishing, 2020.

[12] https://en.wikipedia.org/wiki/Convex_hull. Last accessed: 04.15.23

[13] https://web.stanford.edu/class/archive/ee/ee392n/ee392n.1146/lecture/may13/EE392n_TDA_online.pdf. Last accessed: 04.15.23

[14] Lum, P., Singh, G., Lehman, A. et al. Extracting insights from the shape of complex data using topology. Sci Rep 3, 1236 (2013). https://doi.org/10.1038/srep01236

[15] https://www.weforum.org/agenda/2015/08/how-understanding-the-shape-of-data-could-change-our-world/. Last accessed: 04.15.23

[16] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.ConvexHull.html. Last accessed: 04.15.23

[17] http://jwilson.coe.uga.edu/EMT668/EMT668.Folders.F97/Patterson/EMT%20669/centroid%20of%20quad/ Centroid.html. Last accessed: 04.15.23.

VI. Appendix