CSC 591: Automated Software Engineering Final Project

Dhruvish Patel
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: dmpatel4@ncsu.edu

Urmi Kashyap Pathak
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: upathak@ncsu.edu

Aadil Tajani
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: atajni@ncsu.edu

Sukruthi Modem
Department of Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: smodem@ncsu.edu

***Abstract*—This report explains the Pareto frontier, which represents the set of all possible solutions that are optimal in terms of one objective without compromising the performance of another objective. Additionally, the article discusses the use of multi-objective clustering algorithms on Y to identify different groups of goals with similar trade-offs between the objectives with minimal sampling and generate good learners z.**

I. Introduction

Finding the best mapping function (F) to determine the decisions that will help us achieve our goals is frequently a difficult task. The mapping of inputs (X) to goals (Y) is a fundamental problem in many fields. In a typical scenario, X stands for a set of decisions that we can influence, and Y stands for the results or goals we want to achieve. But because there are typically fewer goals (Y) than decisions (X), this asymmetry creates a challenge for optimization. For instance, when developing software, we might want to balance cost and software quality. Software reliability, performance, maintainability, and cost-effectiveness are among the objectives we hope to achieve (Y), while the decisions we can influence (X) include the choice of programming language, development methodology, and testing strategy. However, clustering Y can offer additional insights into the optimization problem, particularly in identifying various groups of goals with comparable trade-offs between the objectives. Clustering algorithms are frequently used to gain insights from X. The Pareto frontier, which represents the set of all solutions that are ideal in terms of one objective without degrading the performance of another, is one well-known method for solving this type of optimization problem - [6].

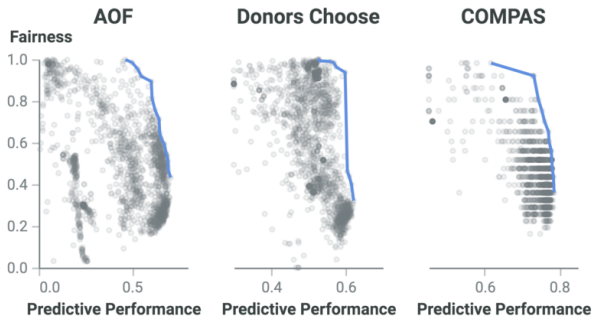

Fig. 1: Pareto Frontier -[9]

We might miss out on some crucial insights that could result in an inadequate mapping function F if we only take into account X and ignore Y. Without taking into account the results or goals we want to achieve (Y), we might optimize for X in a way that is at odds with our overall objectives or, worse yet, have unintended negative effects. The potential for bias in the mapping function F is a problem with only taking into account X, in addition to the danger of less-than-ideal solutions. Our mapping function might favor some decision options while ignoring others because X represents the decisions that we have control over. This may result in a distorted viewpoint and constrict the range of feasible solutions. For instance, when the development team only takes into account X, such as the technology stack or development methodology, without taking into account the inclusivity and diversity of the software user base, bias can occur. This may result in a product that ignores the needs and preferences of other users in favor of only serving the needs of a select group of users. By clustering Y and taking into account all the important goals, including potential biases and the various needs of various user groups, we can make sure that our mapping function F is impartial and fair and offers the best solutions that support our overall objectives.

Our proposed approach not only addresses the issue of bias but also provides a novel insight into the optimization problem by leveraging both X and Y. By clustering Y and analyzing the trade-offs between different objectives, we can identify the Pareto frontier, which represents the set of optimal solutions that balance these objectives. This allows us to make informed decisions that consider all critical objectives and avoid suboptimal solutions that only focus on a single objective. For instance, in software development, if we only optimize for X (e.g., using a specific programming language), we might miss out on critical insights that impact our Y (e.g., software quality and cost). By clustering Y and identifying the Pareto frontier, we can identify the set of optimal solutions that balance these objectives and provide us with a comprehensive view of the trade-offs between different objectives.

Our experimental findings show how well our suggested strategy works to enhance the mapping function F for X and Y. We were able to generate a set of optimal solutions that balanced the trade-offs between various objectives, leading to better decisions for achieving our goals, by clustering Y and locating the Pareto frontier-[2]. Our method also revealed previously hidden trade-offs between goals, offering a fresh perspective on the issue that was impossible to obtain by focusing solely on X. In addition to X and Y, we think that our approach has the potential to advance a number of different fields, and we eagerly await further study and applications of it.

A. Structure

1. How does the asymmetry between X and Y pose an optimization problem?
- Finding the best mapping function (F) that identifies the choices that will help us achieve our goals is known as the mapping of inputs (X) to goals (Y) challenge. Because weighing the trade-offs between various objectives enables us to make informed decisions, this is a fundamental issue in many fields.
2. How does the asymmetry between X and Y pose an optimization problem?
- Finding the best mapping function (F) that identifies the choices that will help us achieve our goals is known as the mapping of inputs (X) to goals (Y) challenge. Because weighing the trade-offs between various objectives enables us to make informed decisions, this is a fundamental issue in many fields.
3. How can clustering algorithms be used to gain insights from X and Y?
- By identifying various groups of decisions with comparable traits, clustering algorithms can be used to extract insights from X. By identifying various groups of goals with comparable trade-offs between the objectives, clustering Y can similarly reveal new information about the optimization problem.
4. What is the Pareto frontier and how can it be used to solve optimization problems?
- The Pareto frontier represents the set of all possible solutions that are optimal in terms of one objective without compromising the performance of another. This approach can be used to solve optimization problems by identifying the trade-offs between different objectives and generating a set of optimal solutions that balance these objectives.
5. How can clustering Y help identify previously unknown trade-offs between objectives?
- Clustering Y can help identify previously unknown trade-offs between objectives by grouping similar goals and analyzing the trade-offs between them. This provides a more comprehensive understanding of the optimization problem and can help identify suboptimal solutions that only focus on a single objective.

Our strategy for the baseline is to lay a strong foundation on which we can build creative solutions that work. We can pinpoint areas where the current system needs to be improved and work to accentuate its advantages by carefully analyzing it. With this strategy, we can make use of the expertise and knowledge accumulated through earlier endeavors while also introducing fresh concepts and methods to advance. By establishing a strong foundation, we can create solutions that are effective and efficient, delivering quantifiable outcomes that satisfy the needs of our clients and stakeholders.

We concentrated on improving the mapping function F in our study, which connects the input variables X to the output goals Y. A set of Z parameters that regulate the learners used in the optimization process produce the mapping function. Due to the nature of the issue, we chose not to investigate how changing these parameters Z would affect the effectiveness of our suggested strategy. Instead, in order to create a set of ideal solutions that balance the trade-offs between various goals, we decided to concentrate on clustering Y and locating the Pareto frontier. Our study lays a solid groundwork for future research to investigate the effect of Z on the optimization problem and the efficacy of our approach in various contexts, even though the effect of Z on the performance of the approach is still an open question.

II. Related Work

Various methods have been proposed in the past to address the multi-objective semi-supervised explanation problem. One common approach is to use model-agnostic methods such as LIME-[1] and SHAP-[8], which can provide local explanations for any black-box model. However, these methods have a high sample and explanation tax and can be computationally expensive, which may restrict their usefulness. Another approach is to use clustering-based methods such as KD-trees-[3] and mini-batch K-means, which have a low sampling tax but a high explanation tax and may not work well in a semi-supervised setting. Furthermore, mini-batch K-means algorithm may converge to a suboptimal solution and have little or no effect on the clustering results if a large set of data points are assigned to a centroid.

The sway is a clustering algorithm that iteratively divides a dataset/cluster into smaller subsets or clusters based on the distance of the data points from two selected points A and B that are the most distant from each other. The algorithm starts with the whole dataset and selects a new pair of points A and B that are the most distant from each other in the current subset of data points. Then, it divides the subset into two halves based on the distance of the data points from A and B. This process is repeated recursively on each half until the subsets become small enough to be considered as clusters. The function takes two arguments: a string indicating which version of the algorithm to use and the dataset to cluster. The output of the function is the best clusters, the remaining data points that were not included in the best cluster, and the number of evaluations performed by the algorithm. The xpln function is used to perform discretization on the clustered data obtained from the sway algorithm. The function takes three arguments: the original dataset, the best clusters obtained from the sway algorithm, and the remaining data points that were not included in the best clusters. It first extracts a decision rule from the best clusters using the xpln function, which is used to perform a modified form of decision tree learning on the best cluster. The decision rule is a set of ranges of values for each variable in the dataset that correspond to a specific outcome or classification. Next, the data points are selected such that they satisfy the decision rule and a new dataset is returned that only contains those data points. This new dataset is then used for further analysis. The xpln function helps in reducing the complexity of the data by discretizing it and also helps in selecting the most important variables and their corresponding ranges.

In the above approach, point X is randomly selected from the available samples and other points A and B are selected based on their distance from A such that their distance is maximum from X. This approach may not always provide the best points that cover the most variation from the available samples and does not ensure that the distance between the two points is maximum every time (example: the random point picked could be the center of the circle, and the most distant point is any point on the diameter of the circle). An improvement to the Sway clustering algorithm that we thought would work better is selecting the initial points A and B to cover the maximum distance between data points, inspired by the PCA (Principal Component Analysis) technique (Figure 2). Our new approach is to create a hypothetical point that is close to our goal. We then select the nearest point to the hypothetical point as point A and find a point that has the maximum distance from point A. This ensures that the selected point A is always on the boundary of the column and that we find the most distant points every time. This strategy may produce clustering results by providing a more representative set of

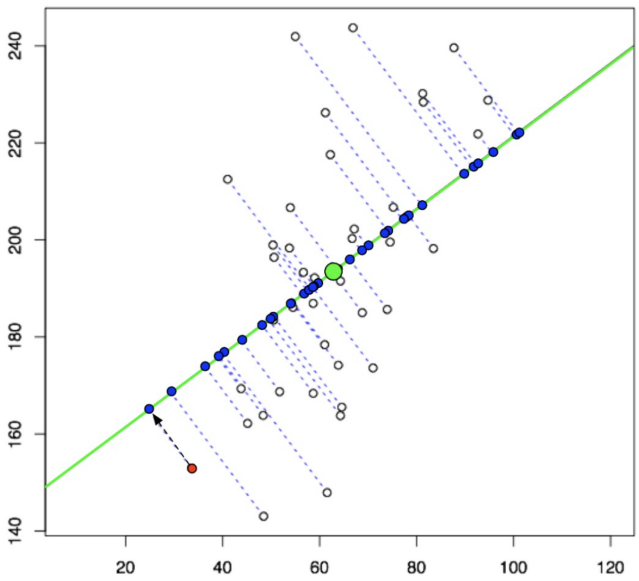

Fig. 2: Principal Component Analysis

initial points for the algorithm. Also, in the XPLN function, we are using ranges that are not useful, which could negatively impact the clustering performance.

III. Methods

A. Algorithms

The FastMap-[7] algorithm is a commonly used method for dimensionality reduction of high-dimensional data. The algorithm first identifies two anchor points in the dataset that are farthest apart from each other, which are used to create a one-dimensional space onto which all other points are projected. The cosine similarity function is used to calculate the distance between any two points in the dataset. This distance is then used to project each data point onto the one-dimensional space by computing its distance to the two anchor points and projecting it onto the line connecting them.

In our proposed approach, we introduce a modification to the baseline approach. Specifically, instead of selecting the first point as a random point from the dataset, we consider a hypothetical point outside of the data points and calculate a point A from this hypothetical point. This ensures that the points A and B are the farthest from each other. The figure below illustrates this approach. This modification results in a more optimal choice of anchor points.

Fig. 3: Finding A from Hypothetical point H

The changes to the code are made to the 'half' function which divides a set of data into two halves based on the distance between

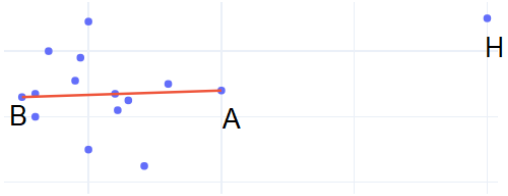

Fig. 4: Finding the furthest point B from A

two anchor points. Finally, the 'cluster' function recursively divides the dataset into smaller subsets using the half function, resulting in a hierarchical clustering of the data.

```
newA = []
for j in data.cols.all: //for all columns
   i = j.col
   if i.txt[-1] in "X": //ignore column ends in X
        newA.append(None)
   elif i.txt[0].strip().isupper(): //if numeric
      mu = abs(i.hi - i.lo)
      if i.txt[-1] in "+-!": //if goal
        newA.append(i.lo-mu if i.txt[-1]=="-"
            else i.hi+mu) //set hypothetical
            column value to low-range if min goal
            else hi+range if max goal
      else:
        rdata = any(i.has)
        newA.append(rdata) //else add any data
            from available data
   else: //if symbolic add any data from
       available data
      newA.append(any(list(i.has.keys())))


tmp1 = sorted([{"row": r, "d": gap(r, [str(i)
    for i in newA])} for r in some], key=lambda
    x: x["d"])
A = above or tmp1[0]['row']
```

One of the key modifications we made to the baseline algorithm was the introduction of the xpln rule generator, which leverages the best and rest values to generate effective learners z, which can be used to cluster the entire dataset and achieve the goals. The original baseline algorithm used a function called 'useful' to identify the bins that maximized the goals by filtering based on a specific threshold value. However, we proposed an approach to obtain the leftover ranges that exclusively produced rest values by using the neg function to identify half of the most useful values that only resulted in rest values. These values were then passed to the score function to generate another range.

```
def useful(range): //get useful ranges
      if range["val"] > 0.05 and range["val"] >
          first / 10:
         return range

def neg(range): //get other excluded ranges
   if range["val"] < 0.05 and range["val"] <
       first / 10:
      return range

negranges = list(filter(neg, sortedRanges))
negranges.reverse()
```

```
sortedRanges = list(filter(useful, sortedRanges))
most, out = -1, None

for n in range(len(sortedRanges)):
   tmp, rule = scoreFun([r["range"] for r in
       sortedRanges[:n + 1]], [r["range"] for r
       in negranges[:floor(len(negranges)/2)]])
       or (-math.inf, None) //pass half of neg
       ranges to score function
```

The proposed modification aims to address the potential ambiguity of the baseline method in classifying the best values by considering the possibility of other variables leading to some rest values. To achieve this, the modified method checks whether a data point satisfies the useful range and also does not match the neg range, which is more likely to produce rest values. This approach aims to provide more accurate classification of the best values in the data.

```
def conjunction(row):
      if rule:
         for ranges in rule['pos'].values():
            if ranges:
               for neg in rule['neg'].values():
                  if neg: //if neg ranges
                      available
                     if disjunction(neg, row):
                        return False //data
                            should not match neg
                            range
               if not disjunction(ranges, row):
                  return False //data should
                      match the useful range
         return True
      else:
         return True

   return [r for r in rows if conjunction(r)]
```

B. Data

Auto2.csv

This data is multivariate, and has 93 instances, and 23 attributes. The domain of this dataset is car design and gives information about several in-depth car features such as car manufacturer type, airbags standard, engine size, and more.

Auto93.csv

This data is multivariate, and has 398 instances, and 8 attributes. The domain of this dataset is car design and gives information about no. of cylinders, volume, model and many more.

China.csv

This dataset contains 499 instances and 19 attributes and gives information regarding software project estimation like IDX, AFP, Enquiry, Interface.

Coc1000.csv

This dataset contains 1,000 numerical instances and 25 attributes and gives information regarding software project estimation like risk, effort, loc.

Coc10000.csv

This dataset contains 10,000 numerical instances and 25 attributes and gives information regarding software project estimation like risk, effort, loc.

Nasa93dem.csv

This dataset contains 93 instances and 29 attributes which comprise of numerical and symbolic data representing software efforts and detect estimation.

HealthCloseIsses12mths0001-hard.csv

This dataset contains 10,000 instances and 8 attributes which comprise of numerical and symbolic data representing software efforts and detect estimation.

HealthCloseIsses12mths0011-easy.csv

This dataset contains 10,000 instances and 8 attributes which comprise of numerical and symbolic data representing software efforts and detect estimation.

Pom.csv

This dataset contains 10,000 instances and 13 attributes which gives information on agile project management by explaining data with criticality, interdependency, plan and many more.

SSM.csv

This dataset contains 2,39,360 instances and 15 attributes which gives information on computational physics by explaining data with presmoothing, alpha, beta and many more.

SSN.csv

This dataset contains 53,662 instances and 19 attributes which gives information on computational physics by explaining data with no. of threads, energy, PSNR and many more.

TABLE I: data comparison

Data	#X	#Y	#rows
auto2.csv	19	4	93
auto93.csv	4	3	398
china.csv	17	1	499
coc1000.csv	20	5	1000
coc10000	20	5	10000
nasa93dem.csv	22	4	93
healthCloseIsses12mths0001-hard.csv	5	3	10000
healthCloseIsses12mths0011-easy.csv	5	3	10000
pom.csv	11	3	10000
SSM.csv	13	2	239360
SSN.csv	17	2	53662

Here #X means no. of independent variables, #Y means no.of goals or dependent variables and #rows is the no. of rows in the dataset

C. Performance measures

Performance measures for the experiment were used as Cliff's Delta effect size test which shows if the results are significantly different or not by comparing both distributions which tells us if we have made a significant change or not which is shown in the figure. Upon looking at performance, we carried out an ablation study to see if the algorithm was executed without the selection of A by our method, and yielded drastic results denoting its importance of it. As well as, in the distance function which utilizes zitler's continuous predicate for distance calculation and comparing data, when replaced with a euclidean distance function, was not able to cluster well again showing it's importance with the continuous domination.

D. Summarization Methods

We utilized two methods for summarizing the experimental results. Firstly, the means of the best values over 20 repeated runs were calculated using different seed values and presented in a table for comparison with other methods. Additionally, Scott's Knot method was used to summarize the results, which involves using tiles and bootstrapping to show the data spread and central tendency. The tiles function was used to visualize the data with its medians for easy interpretation.

```
=============================================
Effect Size Test Comparison - Cliff's Delta
=============================================
                  MRE-    ACC+     PRED40+
all to all         =       =        =
all to sway1       ≠       ≠        =
all to sway2       ≠       ≠        =
sway1 to sway2     =       =        =
sway1 to xpln1     =       =        =
sway2 to xpln2     =       =        =
sway1 to top       ≠       ≠        ≠
```

Fig. 5: Performance shown by Cliffs Delta

```
============================================
Dataset: healthCloseIsses12mths0001-hard.csv
============================================
Mean results of best outcomes from 20 runs
============================================

         MRE-   ACC+    PRED40+
all      75.12  7.17    25.0
sway1    73.88  7.53    23.75
xpln1    73.77  7.54    25.0
sway2    73.72  7.66    25.0
xpln2    73.78  7.58    25.0
top      64.18  33.03   37.5
```

Fig. 6: Summary of data over 20 repeated runs

IV. Results

As discussed above, we have tried augmenting different changes to try and get better clusters as well as a better explanation of the data in order for rule generation, which upon mapping, would provide us with the best data points by looking at only a few points - the task at hand. The results obtained from the solution for the dataset 'healthCloseIsses12mths0001-hard.csv' is as follows when executed with different sampling budget sizes in the order of 20, 50, 128, 512, and 1024. The values obtained are the mean values from 20 repeated runs with different seed values. The results and comparison of our methods(sway2 and xpln2) and prior state-of-the-art methods(sway1 and xpln1) are shown for the goals: MRE-, ACC+, and PRED40+. The results show that our methods sway2 and particularly xpln2 are performing marginally better at budget size 512. This was observed for most of the datasets where budget size 512 yielded the best results out of our methods and needs to be further explored. Although the new methods perform marginally better, stats of effect size test comparison from Cliff'sDelta shows the difference between the distribution of results are insignificant when compared as shown in table 4.1.

In addition to the baseline algorithm, we have explored several attempts to improve its performance. Specifically, we have experimented with two approaches: (1) reusing the A value for distance

```
ScottsKnot for: ACC+
        1       all      -*-              |                        {  7.10,   7.13,   7.18,   7.21,   7.24}
        2       sway1         --*---      |                        {  7.44,   7.49,   7.52,   7.56,   7.68}
        3       sway2        --- * -------|---                     {  7.41,   7.51,   7.58,   7.66,   8.10}
        3       xpln1          -*-        |                        {  7.50,   7.52,   7.55,   7.57,   7.59}
        4       xpln2          --*----    |                        {  7.50,   7.55,   7.59,   7.61,   7.76}
```

Fig. 7: ScottsKnot shown using tiles to summarize results

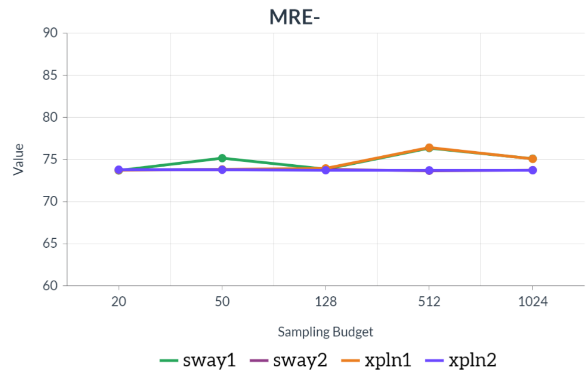

Fig. 8: Sampling budget comparison for MRE-

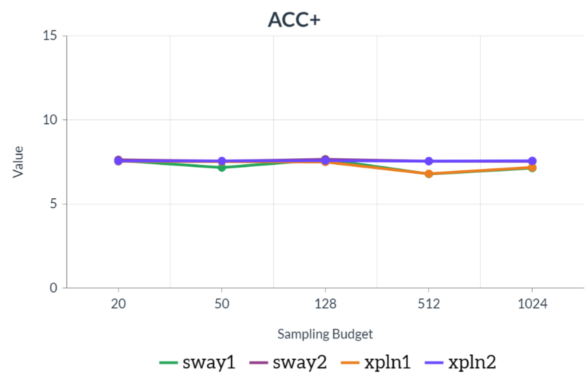

Fig. 9: Sampling budget comparison for ACC+

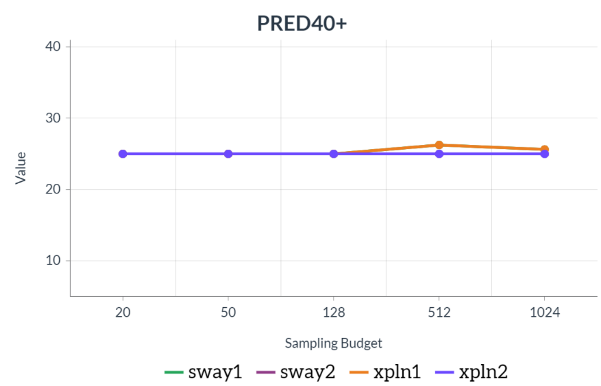

Fig. 10: Sampling budget comparison for PRED40+

TABLE II: effect size test comparison by cliff's delta

Comparison	MRE-	ACC+	PRED40+
All to All	=	=	=
All to Sway1	$\neq$	$\neq$	=
All to Sway2	$\neq$	$\neq$	=
Sway1 to Sway2	=	=	=
Sway1 to Xpln1	=	=	=
Sway2 to Xpln2	=	=	=
Sway1 to Top	$\neq$	$\neq$	$\neq$

endpoint from a parent (2) generating the A value from the hypothetical point each run, and (3) incorporating a multiplication factor into Zitler's continuous domination predicate to provide dynamic weighting to differences based on their data ranges [4].

The first approach involves leveraging the fact that, in some cases, the distance endpoint for a given point may be the same as that of its parent. By reusing the A value in such cases, we can reduce the number of computations required and potentially improve the algorithm's efficiency. The second approach involves generating the A value for each hypothetical point on-the-fly, rather than relying on a precomputed value. This can be advantageous in situations where the data is irregular, as it allows the algorithm to more accurately capture the characteristics of the spread. Adding a multiplication factor was proposed to introduce dynamic weighting of goals. Specifically, this factor amplifies the importance of differences where the data range for a particular goal is small, in order to highlight the significance of changes where the range is already small compared to other goals.

The differences are shown in the figure below where it can be seen that reusing the A value provided better results in both sway and xpln, whereas adding the multiplication factor didn't create any difference.

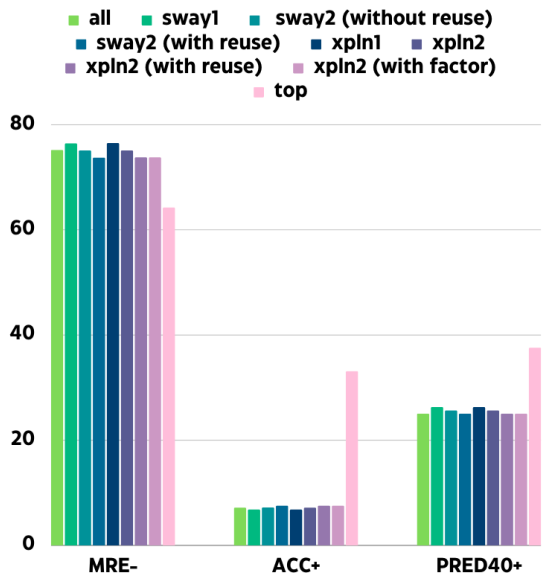

Fig. 11: Method comparison results

During our initial testing, we observed that the second half of the data provided better coverage of the variance in the data for rule generation. We subsequently applied our algorithm with a sampling budget of 400 solely to the second half of the data, and found that the algorithm performed better. This allowed us to leverage the knowledge gained from our initial tests and apply it in subsequent studies, such as the February study shown in the tables 3 and 4, with even smaller budgets. This approach enabled us to optimize our algorithm and achieve better performance results.

TABLE III: Initial study with budget 512

Type	MRE-	ACC+	PRED40+
all	75.12	7.17	25.0
sway1	76.36	6.79	26.25
xpln1	76.42	6.8	26.25
sway2	73.67	7.55	25.0
xpln2	73.72	7.55	25.0
top	64.18	33.03	37.5

TABLE IV: Second study with half data and budget 400

Type	MRE-	ACC+	PRED40+
all	75.12	7.17	25.0
sway1	73.88	7.53	23.75
xpln1	73.77	7.54	25.0
sway2	73.72	7.66	25.0
xpln2	73.78	7.58	25.0
top	64.18	33.03	37.5

V. Discussion

Our experiments have revealed some interesting findings regarding the performance of the proposed solution. Firstly, we observed that the algorithm achieved the best results with a sampling budget of around 512, neither less nor more, indicating a fit to the sample size perhaps which needs to be explored. Secondly, the modified solution appeared to perform marginally better on certain datasets, such as healthCloseIsses12mths0001-hard, healthCloseIsses12mths0011-easy.csv, auto93, and china, while performing slightly worse on others, such as nasa93dem, coc1000, coc10000, pom, and SSN showing strains over datasets of different sizes and goals.

Moreover, our analysis suggests that the performance of the modified solution may be influenced by the difference in ranges among the goals within a given dataset. Specifically, the algorithm appeared to perform better for minimization goals as compared to maximizing goals.

Although the multiplication factor for Zitler's continuous domination predicate did not yield any noticeable change in performance, it may be worthwhile to further explore alternative approaches to optimizing this factor in future research[5].

VI. Conclusion

After conducting extensive modifications to algorithms and tests on various datasets with different seed values, our findings indicate that the modified fishing algorithm performs comparably to the baseline algorithm, with some marginal improvements observed in certain cases, learning good z values for a learner using minimal sampling methods, showing the HPO study. Results from the effect size test conducted using Cliff's Delta reveal that the difference in output distribution between both algorithms is negligible, making it challenging to determine which algorithm performs better overall. We conclude that the performance difference is heavily influenced by factors such as the seed values, distance functions, and dataset characteristics, such as length, range of goals, and spread of data. Our study suggests that further improvements to the modified fishing algorithm could be achieved by exploring the issues highlighted in our discussions as future work.

Acknowledgment

We would like to express our gratitude to the faculty and staff of the Department of Computer Science at North Carolina State University, and particularly to our instructor Dr. Timothy Menzies for CSC 591 Automated Software Engineering. Their guidance, support,

and encouragement were instrumental in the successful completion of this research paper.

We would also like to acknowledge the valuable feedback and insights provided by our Teaching Assistants throughout the semester on homeworks, concepts and doubts, which helped to refine our understanding of the subject matter and strengthen our research.

Finally, we extend our sincere thanks to the open-source community for providing access to the tools and resources that enabled us to conduct our experiments and analysis.

REFERENCES

[1] Nesaretnam Barr Kumarakulasinghe, Tobias Blomberg, Jintai Liu, Alexandra Saraiva Leao, and Panagiotis Papapetrou. Evaluating local interpretable model-agnostic explanations on clinical machine learning classification models. pages 7–12, 2020.

[2] Lilian Bejarano, Helbert Espitia, and Carlos Montenegro. Clustering analysis for the pareto optimal front in multi-objective optimization. *Computation*, 10:37, 03 2022.

[3] Yu Cao, Xiaojiang Zhang, Boheng Duan, Wenjing Zhao, and Huizan Wang. An improved method to build the kd tree based on presorted results. pages 71–75, 2020.

[4] Dan Chen, Tian Du, Jianwei Zhou, and Li Li. Dwdp-stream: A dynamic weight and density peaks clustering algorithm for data stream. *International Journal of Computational Intelligence Systems*, 15(1):96, 2022.

[5] Si-Bao Chen, Hai-Xian Wang, and Bin Luo. On dynamic weighting of data in clustering with k-alpha means. pages 774–777, 2010.

[6] Hashem I Muñoz López CA De Buck V, Nimmegeers P and Van Impe J. Exploiting trade-off criteria to improve the efficiency of genetic multi-objective optimisation algorithms. *Chem. Eng. 3:582123.*, 2021.

[7] Christos Faloutsos and King-Ip Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *ACM international conference on management of data (SIGMOD)*, 2, 06 1997.

[8] Chejarla Santosh Kumar, Movva Naga Sumanth Choudary, Vinay Babu Bommineni, Grandhi Tarun, and T Anjali. Dimensionality reduction based on shap analysis: A simple and trustworthy approach. pages 558–560, 2020.

[9] Tim Menzies. Ase23. https://github.com/txt/ase23, 2023.