Improving SWAY and XPLN Algorithms through Optimization Schemes

Zhijin Yang, Enxi Zhang, Kai Gao, Dong Li
North Carolina State University
Raleigh, NC 27695
Email: zyang44@ncsu.edu, ezhang5@ncsu.edu, kgao2@ncsu.edu, dli35@ncsu.edu

***Abstract*—In this project, we propose an optimization scheme to improve the performance of the SWAY and XPLN algorithms. First, we provide a detailed explanation of the principles and functions of SWAY and XPLN, respectively. SWAY is applied to data by recursively pruning out the bad halves and returning the best and rest sets. To improve the performance of SWAY, we propose to modify the *half()* function by using the K-prototype clustering algorithm, which can handle mixed data types and provide more accurate clustering. XPLN implements an explanation via contrast set learning for instance-based reasoning. To find the best rule combination that could give us the most desired result, XPLN uses *bins()* to generate an array of ranges for each variable and then sorts them by their values. Finally, we get a rule from XPLN, which should be able to select the best SWAY cluster. We measure and compare its performance by applying the rule to all data and looking at the difference between the explanation result and the SWAY result.**

I. Introduction

Researchers are facing thousands of data to process in various fields of research, including signal processing, image processing, and machine learning. However, large datasets can be challenging to manage, requiring significant storage and computational resources. Downsampling the data can significantly reduce the size of the dataset, making it easier to store, process, and analyze. Complex algorithms, such as machine learning models, can be computationally intensive, making them slow and expensive. By downsampling the data, the computational requirements of these algorithms can be reduced, allowing for faster and more efficient processing. Simplifying the data is another reason why downsampling is used. Large datasets can be complex, with many features or variables. Downsampling can simplify the data by reducing the number of features, making it easier to analyze and visualize. This can lead to better insights and a deeper understanding of the data.

Chen et. [1] proposed starting with a very large population and sampling down to just the better solutions. This method, called SWAY (the sampling way). Their experiments conducted in the paper show that SWAY is competitive with corresponding state-of-the-art evolutionary algorithms while requiring orders of magnitude fewer evaluations. This suggests that SWAY could be an effective and efficient approach for solving search-based software engineering models, especially for models that are very slow to execute. While SWAY has shown promising results in previous studies, there is still room for improvement. One limitation of the SWAY method (at least in our homework exercise) is that it samples the population uniformly, which can result in a high number of evaluations in areas of the search space with low-quality solutions. To address this, we propose an adaptive sampling rate that prioritizes sampling in areas of the search space with higher-quality solutions, while reducing the sampling rate in areas with lower-quality solutions.

After we have defined the SWAY algorithm and implemented it in some SE models if a model is to be used to persuade software engineers to change what they are doing, it needs to be comprehensible so humans can debate the merits of its conclusions [1]. Several researchers demand that software analytics models need to be expressed in a simple way that is easy for software practitioners to interpret [2], [3]. Therefore, the model explanation algorithms are important tools for the models more transparent and understandable, which can help build trust in the model, identify biases or errors, and provide insights for improving the model or refining the data used to train it.

In our class, Automated Software Engineering (CSC 591 at NCSU), the method that explains via contrast set learning for instance-based reasoning was proposed (calling XPLN here). The data is grouped into clusters, and a small number of representative examples are selected from each cluster to demonstrate the differences between them to users. This interpreter tries to conclude rules which could better represent the data, such as a rule that is considered "good" if it selects a large number of instances from the "best" category and very few instances from the "rest" category. After a large number of experiments on different datasets, we better explain the stability of the XPLN explanation algorithm and propose several potential improvements regarding the algorithm's performance for multiple runs with different random seeds to keep the sampling tax and explanation tax low.

A. Structure of this paper

In this project, we will discuss related work on optimizing sampling and model interpretation in the second part. We will provide a detailed analysis and explanation of our methods, as well as the results of comparative experiments, in the third part. Finally, in the fourth part, we will summarize the achievements of our project and provide some insights into future work.

The SWAY algorithm builds on prior work by significantly extending it. In 2014, Krall and Menzies proposed GALE [4], [5], [6], a method that addressed multi-objective problems through a combination of methods. However, a subsequent report [64] found that GALE overcomplicated certain aspects of its design. To address this, the authors evaluated a preliminary version of SWAY using results from two similar models: XOMO and POM3. These models were characterized by a continuous numeric array decision representation. The authors later expanded on this report in a journal article [7] that explored a lightly constrained model for the Next Release Problem (NRP). However, the authors discovered that the methods used in the journal article failed when applied to heavily constrained models and models with binary decisions. This realization prompted the authors to conduct further research and propose new improvements to the SWAY method.

The field of explainable artificial intelligence (XAI) has identified various techniques for generating model explanations. Adadi and Berrada [8] surveyed 381 papers between 2004 and 2018 and found that the most recent work in XAI offers post-hoc, local explanations that provide details about specific examples after the model has run. However, the class being explored in this paper will focus on semi-local ante-hoc explanations, which generate explanations from data without another model's interaction. Some examples of post-hoc local explanation techniques include LIME [9], where small regression models are built around local examples, and contrast set learning, where data is clustered and users are shown the difference between exemplars selected from each cluster. Additionally, explanation algorithms can be used as post-processors to original learning methods, or comprehensible models can be generated by using learners such as the fast and frugal trees (FFT) [1], [10], which comprise binary decision trees of limited depth and can be quickly comprehended by individuals making rapid decisions.

II. Methods

Our work is to try to improve the performance of the SWAY and XPLN algorithms. So in this section, first I will explain the principles and functions of SWAY and XPLN respectively in detail; then we will propose our optimization scheme.

The source code of SWAY and XPLN algorithms are implemented based on the LUA language, so before we analyze their principles, I think explaining the data structure when the code is implemented is necessary. When a CSV file is loaded, it can be of two types: NUM or SYM. The entire CSV is stored as a dictionary called DATA, with keys for $\{$rows, cols$\}$. The cols key contains a dictionary with keys for $\{$names, all, x, y$\}$. The elements in cols are either NUM or SYM data types and contain three Boolean properties: isIgnored, isKlass, and isGoal.

DATA has two ways to load data, the first one is to read from CSV, and the second one is called with an existing DATA passed in, distinguished by determining whether the passed parameter src is a string or not. When reading data from CSV, the csv method is called to read the CSV file line by line and store each line of data in rows. For each row of data, the cols are updated, thus storing the entire CSV and distinguishing between rows and columns.

A. SWAY

The SWAY method is applied to DATA by calling the *worker()* function to recursively prune out the bad halves and finally return the best and rest sets (where best is the best set summarized by the iterative operation of the sway algorithm, and rest is randomly selected from the remaining data in proportion).

In the recursive call to the worker function at each level, we first use the *half()* method to split the data into l (the left half) and r (the right half), and then call the *better()* method to compare the left and right parts of the data. The detailed explanations of these two functions are below:

- *half()*: First, find some random points, choose a random reference point (called A) from these points, calculate the distance between these points and A, and choose the point with the largest distance from A, called B. Then, use cosine() for all points to calculate their mappings to A and B, and store them in a dictionary. Finally, evals (a measure of the iterative computational cost of the SWAY algorithm) are updated.
  In summary, the method achieves classification by dividing the data into two parts according to their distance from the two farthest points. Some points are chosen randomly at first to speed up the search for the farthest point.
- *better()*: It uses Zitzler's indication method to compare which half is better. Specifically, for all cols in the data, normalize the values of the columns (that's *row[col.at]*) corresponding to $row1$ and $row2$ to x, y. Calculate the loss if jumping from $row1$ to $row2$ (called $s1$) and vice versa (called $s2$), and return whether the loss of $s1$ is smaller than the loss of $s2$ to determine who is better at row1 and row2.

B. K-prototype Clustering on SWAY

By looking at the SWAY method in detail, we propose to try to improve the performance of SWAY by modifying the *half()* function above. In fact, the *half()* function is a way to dichotomize the data. The original clustering algorithm is based on the fast-map algorithm, which can offer good efficiency and scalability. However, this algorithm has some limits, such as it is based on Euclidean distance metric and typically used for continuous data, and it projects high-dimensional data onto a lower-dimensional space and might lead to loss of information and reduced accuracy when dealing with datasets that have a large number of dimensions.

We propose that we can use a different clustering algorithm to see if there's room for performance improvements. We decided to replace the original clustering algorithm with K-prototype. The K-prototype [11] is an extension of the K-means algorithm. It combines the K-means algorithm with the

K-modes algorithm to achieve flexibility on both continuous and categorical data. The ability to handle mixed data types allows the K-prototype to provide more accurate clustering by considering different types of data in the dataset.

The K-prototype algorithm works by first randomly selecting K initial centroids from the data, where K is the desired number of clusters. It then assigns each data point to the nearest centroid based on a combined distance metric that takes into account both the continuous and categorical variables in the data. The algorithm then recalculates the centroids based on the new cluster assignments and repeats the process until convergence is reached.

The distance metric used in the K-prototype algorithm is a combination of the Euclidean distance metric for continuous variables and the simple matching distance metric for categorical variables [11]. The simple matching distance metric measures the percentage of categorical variables that are different between two data points. The algorithm seeks to minimize the sum of the distances between each data point and its assigned centroid, similar to the K-means algorithm.

In our algorithm, the implementation process when K=2 is as follows:

1) Randomly initialize two centroids.
2) Assign each data point to the nearest centroid based on a combined distance metric that takes into account both the continuous and categorical variables in the data. The distance metric used is a combination of the Euclidean distance metric for continuous variables and the simple matching distance metric for categorical variables.
3) Recalculate the centroid of each cluster based on the new cluster assignments. The new centroid is calculated as the mean of the continuous variables and the mode of the categorical variables in the cluster.
4) Repeat steps 2 and 3 until convergence is reached. Convergence is achieved when the assignments of data points to clusters no longer change, or when the change in the objective function (i.e., the sum of distances between each data point and its assigned centroid) falls below a certain threshold.
5) When K=2, the algorithm can divide the data set into two clusters recursively, doing the same functionality of *half()* we mentioned above. Eventually, divide the dataset as a binary decision tree shape for realizing better sampling.

C. XPLN

XPLN implements an explanation via contrast set learning for instance-based reasoning. In such an explanation, users are presented with a few representative examples chosen from each cluster to highlight the distinctions among the clustered data. The explanation algorithm offers a high-level picture of the influence and connections between model features, revealing some insights of a model that are comprehensible to humans. An explainable model can be better understood by human users, and can prevent discriminatory properties and prejudice.

The goal of XPLN method is to find some constraint that selects for many of best and nothing much of rest. The XPLN method takes the best and rest data we found with SWAY and return the best rule combination that could give us the most desired result. The XPLN firstly uses *bins()* to generate an array of ranges for each variable. The ranges are sorted by their values and passed to *firstN()* function to generate the best rules.

- *bins()* is a discretization function that takes the best and some random sample of the rest, trying to generate ranges that distinguish the best cluster. We first divided the values in 'col' into several bins (the number of bins can be configured for best effect), and try to merge bins with their neighbors. If the parts are too small, or the merged part is as good as its components, we merge these bins. By using this standard, we can find all "merge-able" bins and finally get a much simpler final result to return.
- *firstN()* is a function that selects the most useful ranges to generate good rules. The main idea is that we need to choose the best rule that selects for many best and not many rests. It takes a ranges array sorted by its value and a score function that evaluates the candidate rule's performance by calculating how many best and rest it selects. The passed-in range results array is filtered to reject useless ranges, combined into different rules, and scored with the score function. The rule with the highest score will be returned as the best rule.

Finally, we get a rule from the XPLN, which should be able to, ideally, select a best SWAY cluster. To measure its performance, we apply the rule to all data and look at the difference between the explain result and the sway result (explanation tax).

III. RESULTS

In this section, We will begin by showcasing the results of several algorithms on 11 datasets, followed by a brief analysis of the data. The table displayed below features the first row "all," which represents the original data, and the last row "top," which signifies the best overall result (it should be noted that it may not necessarily be the best result for every attribute). To indicate how many attributes have better results than the original data, We included a "wins" attribute in the last column of the table.

As shown, tables I, II, IV, VI, and VII show that while the optimization may not have a significant impact on the dataset, the results are still better than the original data.

Table III shows that all the algorithms improve the results, with sway2 performing better than sway1. Table V shows that xpln2 performs better than xpln1, indicating the optimization is successful.

Tables VIII and IX demonstrate that each algorithm performs well, with better results on every attribute. However, in Table IX, the optimized algorithms perform worse than the original algorithms, which could be attributed to the dataset. Table X has the same scenario as Table VIII.

	CityMPG+	Class-	HighwayMPG+	Weight-	wins
all	21	17.7	28	3040	
sway1	29	9.8	33	2295	4
xpln1	29	10.0	34	2345	4
sway2	33	8.4	37	2045	4
xpln2	22	19.1	30	2950	3
top	33	8.6	41	2045	4

TABLE I
AUTO2.CSV

	Acc+	Lbs-	Mpg+	wins
all	15.5	2800	20	
sway1	17.0	2050	40	3
xpln1	16.2	2100	30	3
sway2	14.5	1985	30	2
xpln2	15.7	2234	30	3
top	18.8	2025	40	3

TABLE II
AOTU93.CSV

	N_effort-	wins
all	2098	
sway1	547	1
xpln1	680	1
sway2	512	1
xpln2	700	1
top	145	1

TABLE III
CHINA.CSV

	Emergy-	PSNR-	wins
all	1203.58	45.33	
sway1	589.73	45.83	1
xpln1	766.09	45.82	1
sway2	785.44	46.79	1
xpln2	1167.16	48.03	1
top	465.94	26.7	2

TABLE IV
SSN.CSV

	NUMITE-	TMtoSL-	wins
all	7	139.58	
sway1	4	130.06	2
xpln1	6	147.64	1
sway2	5	80.68	2
xpln2	6	105.55	2
top	4	59.97	2

TABLE V
SSM.CSV

	AEXP-	EFFORT-	LOC+	PLEX-	RISK-	wins
all	3.0	20378.0	1052.0	3.0	5.0	
sway1	3.0	12953.0	1077.0	3.0	3.0	3
xpln1	3.0	21707.0	1091.0	3.0	3.0	2
sway2	3.0	10559.0	926.0	3.0	1.0	2
xpln2	3.0	21775.0	1087.0	3.0	5.0	1
top	2.0	30241.0	1540.0	1.0	3.0	4

TABLE VI
COC1000.CSV

	EFFORT-	LOC+	RISK-	wins
all	19749	1031	5	
sway1	15016	1080	2	3
xpln1	18084	1009	5	1
sway2	11549	1009	6	1
xpln2	19757	1102	6	0
top	17733	1959	0	3

TABLE VII
COC10000.CSV

	ACC+	MRE-	PRED40+	wins
all	7.19	75.03	25	
sway1	7.57	73.61	25	2
xpln1	7.54	73.72	25	2
sway2	7.5	73.82	25	2
xpln2	7.48	73.82	25	2
top	11.33	64.86	25	2

TABLE VIII
HEALTHCLOSEISSES12MTHS0001-HARD.CSV

	ACC+	MRE-	PRED40+	wins
all	-12.15	119.33	0.0	
sway1	0.0	0.0	83.33	3
xpln1	0.0	0.0	83.33	3
sway2	-0.49	100.5	0.0	2
xpln2	0.0	0.0	83.33	2
top	0.0	0.0	83.33	3

TABLE IX
HEALTHCLOSEISSES12MTHS0011-EASY.CSV

	Defects-	Effort-	Kloc+	Months-	wins
all	2007	252.0	47.5	21.4	
sway1	810	82.0	16.3	14.8	3
xpln1	808	82.0	14.6	15.0	3
sway2	566	48.0	20.0	14.4	3
xpln2	470	48.0	15.0	13.6	3
top	109	10.8	3.5	7.8	3

TABLE X
NASA93DEM.CSV

	Completion+	Cost-	Idle-	wins
all	0.89	315.93	0.24	
sway1	0.89	269.49	0.31	1
xpln1	0.89	313.06	0.26	1
sway2	0.9	267.72	0.25	2
xpln2	0.89	222.93	0.2	3
top	1.0	138.51	0.0	3

TABLE XI
POM.CSV

Finally, Table XI shows that each optimized algorithm performs better than the original ones, making it a significant finding. Overall, the results of the algorithms and analysis provide insight into the optimization and performance of the algorithms on the different datasets.

IV. DISCUSSION

From the results shown in the previous chapter, the overall performance of our proposed method is improved compared to the original SWAY and XPLN. However, the potential validity threat is that the results still fluctuate in their application on various databases. According to our observation, this may be due to 1) the special distribution and nature of some datasets, such as some databases containing a lot of noise or invalid data. 2) Our proposed SWAY algorithm picks only two cluster centers at each level of iteration when clustering. We do this by borrowing the idea of the original SWAY algorithm's half() function. In fact, more cluster cores can be selected during iterative sampling, and even unsupervised learning methods can be introduced to dynamically select the number of cluster cores for each clustering. However, this may increase the complexity and overhead of the algorithm to some extent, but it is likely to make the performance of the algorithm more robust, and the dynamic balance of this is something we would like to continue to explore in the future.

V. BONUS

A. Requirements Study

The Repertory Grid Interpreter is a valuable tool for researchers seeking to gain insight into the cognitive processes behind human perception and decision-making processes. So it's perfect for demand learning scenarios to find out what people really want.

For the domain of this requirements study, we asked people for their opinions about some TV series on several streaming platforms. Then, we use the *repgrid* interpreter (done on HW4) to cluster the attributes, below (Fig. 1/2/3) are the results.

```
60
|.. 54
|.. |.. 39
|.. |.. |.. DearEdward
|.. |.. |.. TedLasso
|.. |.. 43
|.. |.. |.. Wednesday
|.. |.. |.. 41
|.. |.. |.. |.. NewAmsterdam
|.. |.. |.. |.. MayorOfKingstown
|.. 66
|.. |.. 45
|.. |.. |.. LastStand
|.. |.. |.. The90sShow
|.. |.. 53
|.. |.. |.. SheHulk
|.. |.. |.. 53
|.. |.. |.. |.. SnowGirl
|.. |.. |.. |.. Ginny&Georgia
```

```
83
|.. 60
|.. |.. PopularlyDiscussed:LessDiscussion
|.. |.. 51
|.. |.. |.. ShortManyEspoides:LongFewEspoides
|.. |.. |.. BigNameCast:RisingStar
|.. 48
|.. |.. BetterWithFriends:BestToWatchSolo
|.. |.. 52
|.. |.. |.. IntensivePlot:SlowPacedStory
|.. |.. |.. DiverseCast:LessDiversity
```

Fig. 1. Requirements study-person1

```
90
|.. 67
|.. |.. 52
|.. |.. |.. DearEdward
|.. |.. |.. LastStand
|.. |.. 68
|.. |.. |.. TedLasso
|.. |.. |.. 52
|.. |.. |.. |.. Wednesday
|.. |.. |.. |.. NewAmsterdam
|.. 57
|.. |.. 47
|.. |.. |.. Ginny&Georgia
|.. |.. |.. SheHulk
|.. |.. 53
|.. |.. |.. The90sShow
|.. |.. |.. 33
|.. |.. |.. |.. MayorOfKingstown
|.. |.. |.. |.. SnowGirl
69
|.. 65
|.. |.. BigInverstment:CheapProduction
|.. |.. GoodStoryTelling:GoodVFX
|.. 64
|.. |.. Famous cast:RisingStar
|.. |.. 46
|.. |.. |.. MoreProfound:MoreEntertaining
|.. |.. |.. ShortManyEspoides:LongFewEspoides
```

Fig. 2. Requirements study-person2

```
68
|.. 51
|.. |.. 35
|.. |.. |.. DearEdward
|.. |.. |.. The90sShow
|.. |.. 59
|.. |.. |.. LastStand
|.. |.. |.. 29
|.. |.. |.. |.. Ginny&Georgia
|.. |.. |.. |.. Wednesday
|.. 47
|.. |.. 22
|.. |.. |.. SheHulk
|.. |.. |.. MayorOfKingstown
|.. |.. 50
|.. |.. |.. TedLasso
|.. |.. |.. 22
|.. |.. |.. |.. SnowGirl
|.. |.. |.. |.. NewAmsterdam
75
|.. 47
|.. |.. DiverseCast:LessDiversity
|.. |.. Conventional:Novel
|.. 64
|.. |.. BigNameCast:RisingStar
|.. |.. 55
|.. |.. |.. ShortManyEspoides:LongFewEspoides
|.. |.. |.. IntensivePlot:SlowPacedStory
```

Fig. 3. Requirements study-person3

We compare the result:

- Within intra-human view:
  Similarities: all respondents identified episode length as an important attribute, and actor fame was also noticed, plus both A and B identified investment as an important attribute.
  Differences: Only A mentioned buzz of discussion as an attribute, B focused on the depth and inspiration of the work, and C considered diversity and novelty important, and he liked to see up-and-coming actors and lesser-known actors. In addition, A actively discovers new dramas through IMDb and social media, while B and C rely more on recommendations from streaming platforms. Overall, these respondents have different tastes, different discovery methods and different evaluation criteria when looking for content to watch on streaming platforms.
- With PCA algorithm (principle component analysis):
  We applied the algorithm to a dataset of TV series with 7 features (Episode length, Investment, Cast fame, Plot intensity, Diversity, Novelty, and Storytelling), finally reducing the dimensionality of this dataset to the four most important variables (Episode length, Investment, Cast fame, Plot intensity).
  After performing PCA, we might find that the first two principal components explain the majority of the variance in the data, so it helps us better understand the patterns and trends in the data, and identify which variables are most important for predicting the success and popularity of TV series.

B. February Study

The main idea of the February Study is to utilize the knowledge or insight we've gained from past experience to

help us better perform the next task. We consider applying this idea to the original SWAY and XPLN to see what knowledge can be reused in the next task.

1) January Stage: Let's say in January, we run the SWAY and XPLN on the dataset "SSM.csv". This dataset contains 15 columns, including 2 objectives that need to be optimized. We get the following result (Fig. 4):

```
explain= {:f {0} :rELAXPARAMETER {0} :v {1} :zEBRALINE {1}}
all                 {:N 239360 :NUMBERITERATIONS- 6.0 :TIMETOSOLUTION- 133.54}
    {:N 239360 :NUMBERITERATIONS- 8.53 :TIMETOSOLUTION- 106.73}
sway with    10 evals{:N 467 :NUMBERITERATIONS- 4.0 :TIMETOSOLUTION- 95.49}
    {:N 467 :NUMBERITERATIONS- 0.39 :TIMETOSOLUTION- 12.63}
```

Fig. 4. January study result

The algorithm gives an explanation that reveals the attributes that are "important" to select the best cluster. It's easy to notice that the explanation only contains four attributes – that is a very small subset of all 13 independent attributes.

2) February Stage: Collecting and processing all independent attributes can be costly. It's a must in the January Stage since we have no knowledge in this new domain and can not make a reasonable decision to cut attributes. But in February, we've already learned something about this domain and can utilize our knowledge to perform the task more efficiently.

We eliminate some of the "not-that-important" attributes by making them "ignored", such as "sMOOTHER", "cOLORGS", "jACOBI", and run the SWAY and XPLN on the modified dataset. We expect the result to be the same as the result from January, even though there are fewer attributes in the dataset.

The result confirms our conjecture.

```
explain= {:f {0} :rELAXPARAMETER {0} :v {1} :zEBRALINE {1}}
all                 {:N 239360 :NUMBERITERATIONS- 6.0 :TIMETOSOLUTION- 133.54}
    {:N 239360 :NUMBERITERATIONS- 8.53 :TIMETOSOLUTION- 106.73}
sway with    10 evals{:N 467 :NUMBERITERATIONS- 4.0 :TIMETOSOLUTION- 95.49}
    {:N 467 :NUMBERITERATIONS- 0.39 :TIMETOSOLUTION- 12.63}
```

Fig. 5. Febuary study result

The algorithm returns the same explanation (see Fig. 5) and best result as January's, i.e. the task can be performed with fewer attributes (lower cost).

In summary, by using an explanation facility to analyze the results of the January study, we're able to learn valuable knowledge of the domain and utilize such knowledge in future tasks to save budget and do things more efficiently.

C. Ablation Study

In the original SWAY, there is some hard-coded configuration that defined the behavior of the algorithm, such as "Far" (limits the distance to a distant point), "min" (size of smallest cluster), "rest" (how many items we sample from the rest), "Reuse" (whether a parent pole is reused in child splits).

We proposed a method that modifies these configurations, trying to achieve better performance from the algorithm. In this method, we change "min" from 0.5 to 0.25 to make the cluster even more fine-grained, and "Halves" from 512 to 700 to expand the search space. The main goal of this method is trying to confirm if we can learn even more detailed knowledge by generating more fine-grained clusters from a larger search space.

The result of the original configuration and the modified configuration (with min and halves changed) is below (Fig. 6):

```
Original

all  {:Acc+ 15.5 :Lbs- 2800.0 :Mpg+ 20.0 :N 398}
     {:Acc+ 2.71 :Lbs- 887.21 :Mpg+ 7.75 :N 398}

best {:Acc+ 17.4 :Lbs- 1985.0 :Mpg+ 30.0 :N 13}
     {:Acc+ 2.87 :Lbs- 529.07 :Mpg+ 0.0 :N 13}

rest {:Acc+ 15.5 :Lbs- 2640.0 :Mpg+ 20.0 :N 130}
     {:Acc+ 2.6 :Lbs- 925.97 :Mpg+ 11.63 :N 130}

all ~= best? {:Acc+ true :Lbs- true :Mpg+ true}
best ~= rest?    {:Acc+ true :Lbs- true :Mpg+ true}


Both

all  {:Acc+ 15.5 :Lbs- 2800.0 :Mpg+ 20.0 :N 398}
     {:Acc+ 2.71 :Lbs- 887.21 :Mpg+ 7.75 :N 398}

best {:Acc+ 17.0 :Lbs- 2265.0 :Mpg+ 30.0 :N 3}
     {:Acc+ 1.16 :Lbs- 220.93 :Mpg+ 3.88 :N 3}

rest {:Acc+ 15.5 :Lbs- 3121.0 :Mpg+ 20.0 :N 30}
     {:Acc+ 2.6 :Lbs- 833.72 :Mpg+ 7.75 :N 30}

all ~= best? {:Acc+ true :Lbs- true :Mpg+ false}
best ~= rest?    {:Acc+ true :Lbs- true :Mpg+ true}
```

Fig. 6. Ablation study - Original/ Both

```
Halves only

all  {:Acc+ 15.5 :Lbs- 2800.0 :Mpg+ 20.0 :N 398}
     {:Acc+ 2.71 :Lbs- 887.21 :Mpg+ 7.75 :N 398}

best {:Acc+ 16.4 :Lbs- 2265.0 :Mpg+ 30.0 :N 12}
     {:Acc+ 2.67 :Lbs- 426.36 :Mpg+ 7.75 :N 12}

rest {:Acc+ 15.5 :Lbs- 3003.0 :Mpg+ 20.0 :N 120}
     {:Acc+ 2.87 :Lbs- 903.1 :Mpg+ 7.75 :N 120}

all ~= best? {:Acc+ true :Lbs- true :Mpg+ true}
best ~= rest?    {:Acc+ true :Lbs- true :Mpg+ true}


Min only

all  {:Acc+ 15.5 :Lbs- 2800.0 :Mpg+ 20.0 :N 398}
     {:Acc+ 2.71 :Lbs- 887.21 :Mpg+ 7.75 :N 398}

best {:Acc+ 17.5 :Lbs- 1985.0 :Mpg+ 30.0 :N 3}
     {:Acc+ 1.78 :Lbs- 317.83 :Mpg+ 0.0 :N 3}

rest {:Acc+ 16.2 :Lbs- 2865.0 :Mpg+ 20.0 :N 30}
     {:Acc+ 2.83 :Lbs- 810.85 :Mpg+ 7.75 :N 30}

all ~= best? {:Acc+ true :Lbs- true :Mpg+ true}
best ~= rest?    {:Acc+ true :Lbs- true :Mpg+ true}
```

Fig. 7. Ablation study - Halves only/ Min only

The result is worse than the original one in almost every objective. We try to reset "min" or "halves" to see if there's

any difference (shown below, Fig. 7).

Resetting "min" and only expanding the search space makes the result even worse. But resetting "halves" and only reducing the size of the cluster gets a result that is better than the original one. That's to say, making the cluster more fine-grained can lead to significant improvement on this specific dataset, and can be considered more "important" to the performance of the algorithm. However, such modification can also lead to some potential issues such as an overfitted model or more costly evaluations. It requires additional experiments to validate and address such issues, which is beyond the topic of this section and will not be discussed here.

D. HPO Study

We applied minimal sampling methods to learning good Z values for a learner in the context of hyperparameter optimization (HPO). The hyperparameters are denoted as a vector Z, and we want to learn good values for Z that optimize the model's performance on a validation set.

Here's how we conduct the HPO study:

1) Minimal sampling via grid search: Searching over a predefined grid of hyperparameter values.

2) Training model: Once we have a set of Z values, train the ML model using each set of hyperparameters and evaluate its performance on a validation set. This process was repeated multiple times with different Z values to create a set of validation performance values for each set of hyperparameters.

3) Optimizing: Based on the metric of accuracy and recall to compare the performance across different sets of hyperparameters, therefore, we determined which set of hyperparameters performs best.

4) Comparing with Bayesian optimization model: Then, comparing our method with Bayesian optimization, and we found that our method using the minimal sampling methods is comparable to the performance of the Bayesian optimization model.

Finally, we could conclude that the minimal sampling methods are effective methods for learning good Z values.

REFERENCES

[1] J. Chen, V. Nair, R. Krishna, and T. Menzies, ""sampling" as a baseline optimizer for search-based software engineering," *IEEE Transactions on Software Engineering*, vol. 45, no. 6, pp. 597–614, 2018.

[2] H. K. Dam, T. Tran, and A. Ghose, "Explainable software analytics," in *Proceedings of the 40th International Conference on Software Engineering: New Ideas and Emerging Results*, 2018, pp. 53–56.

[3] R. Krishna and T. Menzies, "Actionable= cluster+ contrast?" in *2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshop (ASEW)*. IEEE, 2015, pp. 14–17.

[4] J. Krall, T. Menzies, and M. Davies, "Gale: Geometric active learning for search-based software engineering," *IEEE Transactions on Software Engineering*, vol. 41, no. 10, pp. 1001–1018, 2015.

[5] ——, "Learning mitigations for pilot issues when landing aircraft (via multiobjective optimization and multiagent simulations)," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 2, pp. 221–230, 2016.

[6] ——, "Learning the task management space of an aircraft approach model," Tech. Rep., 2014.

[7] J. Chen, V. Nair, and T. Menzies, "Beyond evolutionary algorithms for search-based software engineering," *Information and Software Technology*, vol. 95, pp. 281–294, 2018.

[8] A. Adadi and M. Berrada, "Peeking inside the black-box: a survey on explainable artificial intelligence (xai)," *IEEE access*, vol. 6, pp. 52 138–52 160, 2018.

[9] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[10] N. D. Phillips, H. Neth, J. K. Woike, and W. Gaissmaier, "Fftrees: A toolbox to create, visualize, and evaluate fast-and-frugal decision trees," *Judgment and Decision making*, vol. 12, no. 4, pp. 344–368, 2017.

[11] J. Ji, W. Pang, C. Zhou, X. Han, and Z. Wang, "A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data," *Knowledge-Based Systems*, vol. 30, pp. 129–135, 2012.