Multi-Objective Semi-Supervised Explanation System

Prit Modi
North Carolina State University
Computer Science
Raleigh NC, USA
Email: pmodi2@ncsu.edu

Pathey Shah
North Carolina State University
Computer Science
Raleigh NC, USA
Email: pshah7@ncsu.edu

Nikhil Mehra
North Carolina State University
Computer Science
Raleigh NC, USA
Email: nmehra2@ncsu.edu

Abstract—**This paper presents a clustering-based sampling and explanation system for a multi-objective semi-supervised explanation system. Genetic algorithms-based approaches were state-of-the-art models for multi-objective optimization systems but they are computationally expensive and can't scale well. Later SWAY [2] was introduced which starts with very large populations and samples only better solutions and was much faster than traditional genetic algorithms-based models. In this paper, we are proposing a hybrid agglomerative clustering and PCA-based algorithm, SWAY2, which significantly improves the performance of SWAY while increasing the time complexity of the model to a reasonable level. We have also proposed a new explanation algorithm, XPLN2, that explores more rules than the baseline explanation algorithm, XPLN1.**

I. Introduction

A multi-objective semi-supervised explanation system is an AI-based system that combines multiple goals and explains a particular choice with the help of labeled and unlabeled data. The problem can be divided into two sub-problems; sampling the data points and providing human interpretable explanations for the samples. An explanation system makes AI more transparent, fair, and accountable and can boost the confidence of users. Current approaches for sampling use semi-supervised clustering-based methods. Özyer, Alhajj, and Barker [6] proposed weighted K-Means clustering for multi-objective optimization. There are multi-objective evolution algorithms (MOEA) that can be adapted for the multi-objective optimization problem. NSGA-II Deb et al. [3] uses fast non-dominated sorting with selection operator. Chen et al. [2] proposed SWAY (The Sampling Way) for sampling candidates.

Weighted K-Means clustering [6] and NSGA-II [3] are computationally expensive algorithms that are not well suited for larger datasets and might not scale well. SWAY [2] is a fast sampling algorithm that recursively divides data into half and selects one as the better half and proceeds further. The issue with SWAY is that it only considers one element from each cluster to identify a better cluster, hence it can miss out on good candidate points.

From prior work, we observed that K-Means clustering [6] and NSGA-II[3] based approaches are good at assigning a similar group of points into a single cluster. While SWAY is very good at efficiently dividing data into two halves for larger datasets. SWAY divides data into two halves by using a line between two distant points from the data, In order to improve performance SWAY selects one of the points randomly. Because of this randomness, SWAY can miss out on the best splitting line. PCA(principal component analysis) Wold, Esbensen, and Geladi [10] algorithm which is used for dimensionality reduction calculates principal components in order of the variance they capture from the original data. If we split the data across the first principal component we can improve the performance of SWAY [2] within reasonable time complexity.

We are proposing a hybrid approach for sampling, SWAY2, where we are using agglomerative clustering for smaller data with fewer parameters and PCA-based clustering for larger data with many parameters. The advantage of the hybrid approach is agglomerative clustering performs well on smaller data consuming reasonable amount of time and memory, and PCA-based clustering is efficient for larger data and produces better results than SWAY

We have also proposed a new explanation algorithm, XPLN2, that is inspired by the baseline explanation algorithm, XPLN1. It tries to explore rules that the baseline doesn't and utilizes a higher threshold for the score to select the best rule for that particular problem.

A. *Structure of this Paper*

(RQ1) : How does SWAY2's runtime compares to SWAY [2]? SWAY2 uses the first principal component to split the data into two halves and sample a few random points from both halves to find better halves. For fewer optimization parameters SWAY2's speed is similar to SWAY and as the number of optimization parameters increases its time increases with the order of p^2.

(RQ2): Can SWAY2 handle multiple minimization and maximization objectives with reasonable performance? SWAY2 can perform well with multiple maximized and minimized objectives compared to other MOEAs like NSGA-II [3], from the results we have obtained SWAY2 has improved performance over SWAY [2] in both minimizing and maximizing objectives. But it gives far better performance compared to SWAY when it comes to minimizing objectives.

(RQ3): Can XPLN2 find better combination of rules and keep explanation tax low as compared to baseline explanation algorithm, XPLN1? XPLN2 gave comparable results as compared to baseline explanation algorithm and gave better results for few datasets like healthCloseIsses12mths0001-hard as it was able to find better combinations of rules. Also for multiple runs with multiple random seeds it was able to keep explanation tax (loss seen between SWAY2 algorithm and explanation algorithm) low.

We have introduced a novel hybrid approach to sampling by using recursive agglomerative clustering for smaller data and PCA-based clustering for larger data, recursive agglomerative clustering produces much better results for smaller data but it's not efficient for larger data given the CPU time and memory it takes for sampling, For larger data SWAY [2] is extremely efficient in terms of CPU time. We have improved its sampling performance by splitting data across the first principal component while keeping the CPU time within a reasonable limit.

We have explored agglomerative and spectral clustering-based approaches but because of the higher time and memory requirements of both approaches, we were unable to compare their performance for larger datasets.

II. Related Work

Wold, Esbensen, and Geladi [10] proposed VAGA first multi-objective genetic algorithm, where he introduced the method for finding fit population when there are more than one ways to be considered fit. Then many state-of-the-art genetic algorithm-based multi-objective optimization models like NSGA[8], NSGA-II [3], MEA [7] were developed. NSGA-II improved the time complexity of previous MOEAs. NSGA-II works by first non-dominated sorting of the solution population. Solutions that are not dominated by any other solution are assigned the first rank, solutions that are dominated by only one solution are assigned the second rank, and so on. Then the solution of each rank is sorted using crowding distance. Crowding distance measures the closeness of its neighbors. It then creates a new population of solutions by selecting the best solutions from the previous offspring using fit. The same process is repeated until termination criteria are satisfied. Genetic algorithms use mutation, selection, and crossover operations over 100s of generations. Genetic algorithms are CPU intensive which makes them unfeasible for larger datasets.

Krall, Menzies, and Davies [5] proposed GALE for multi-objective optimization problems which has linear time requirements and comparable performance with MOEAs. GALE solved multi-objective optimization problems using a series of methods. GALE used several heuristics to divide the candidate space, but all the heuristics failed to perform for binary variables. The reason for failure over binary variables was simple. When dividing the space, binary variables fall through the opposite end of the vertex. SWAY was failing when it unnecessarily proposed empty divisions between empty spaces of the vertex. Chen et al. [2] solved this problem by introducing a new method for the binary variable split. Chen et al. [2] also found out that the technique genetic algorithm uses for optimization, mutating 100 individuals for 100 generations; similar results can be obtained by just using one generation of 10,000 randomly generated individuals. This led to a simplistic greedy-based approach SWAY.

SWAY [2] recursively clusters the candidates into two halves and identifies the superior half and drops the other half, this process is repeated until cluster size reaches a threshold. Unlike other genetic algorithms, SWAY does not cluster candidates based on their objectives which requires the evaluation of all candidates and it is computationally expensive. To improve performance SWAY clusters candidates by their decision. It assumes that there is a close correlation between decisions and objectives. Initially SWAY first randomly selects a point from the set of points and finds the farthest point from that random point as the pivot point. It again finds the farthest point from the pivot point and clusters all the points based on their distance to both points. Then it compares both pivot points using the Zitzler indicator[11] to identify the superior cluster. Because of this approach, SWAY is computationally much faster than MOEAs. It uses log-linear time for sampling.

While analyzing SWAY results we observed that the line between two endpoints which SWAY uses as the splitting line might not give best split of data points into clusters, we are proposing a new method where we find the first principal component of the data and use median value of the principal component to split data into two halves. The first principal component in the data represents the line that captures the maximum variance of the data. By splitting the data across this line we can obtain better cluster divisions of the points. Also during evaluations SWAY only considers the two farthest endpoints to identify better clusters. That can sometimes lead to the drop of a good candidate cluster. We are proposing a new method to identify better cluster, where we select equal number of random points from both clusters and compare those points using Zitzler indicator. With both this modification to the original SWAY we were able to generate better samples compared to the original SWAY. As we are using the first principal component, the time required to split n samples with p optimization parameters is in order of $n * p^2$. It takes a reasonable amount of time compared to SWAY for a moderate number of optimization parameters. We also observed that agglomerative clustering outperforms PCA-based clustering due to its memory and time intensive nature we are recommending a hybrid approach to use agglomerative clustering for smaller data and PCA-based clustering for larger data.

III. Methods

A. Overview

Our approach to creating new optimization algorithms involved using the baseline SWAY as the basic structure, and modify its components to create something new. The baseline SWAY mainly consists of two components, a clustering algorithm and an evaluation function. There are several other components to each of these, but we have broadly experimented with these two parts. For the clustering algorithms, we have tried to use only those algorithms where we can control the number of clusters, so that we can easily reuse the baseline structure. We took a similar approach with the explanation algorithm, where we have tried to use the same basic code but made changes to the rule making component and some hyperparameters.

B. Algorithms

1) K-means clustering: K-means[4] is a popular unsupervised machine learning algorithm that is used to partition a dataset into k clusters. The algorithm's objective is to maximize the variance between the clusters while minimizing the variance inside each cluster (measured by the sum of square distances). The algorithm starts by selecting k random points from the dataset as the initial centroids. It then assigns each data point to the closest centroid based on the distance. The distance between two points can be calculated using any distance metric, such as Euclidean distance or Manhattan distance. After all the data points have been assigned to their respective clusters, the centroids are recalculated as the mean of all the data points in the cluster. This process is repeated until the centroids no longer change or a specified number of iterations have been reached. For this project, we have used sklearn's implementation of the k-means algorithm. We have set the number of clusters hyperparamter (k) to be 2 and have left the other hyperparameters to their default values.

2) Agglomerative clustering: Agglomerative clustering is also a popular unsupervised machine learning algorithm used to group similar data points together into clusters. The algorithm starts with each data point as its own cluster. It calculates the distances between each cluster using a distance metric. We have used Euclidean distance in this project. There are different ways to define the distance between two clusters, such as:

- Single linkage: The minimum of the distances between all observations of the two clusters.
- Complete linkage: The maximum distances between all observations of the two clusters.
- Average linkage: The mean of the distances of each observation of the two clusters.
- Ward[9] linkage: Utilizes the optimal value of an objective function.

The two closest clusters are merged using one of these criteria. The distance matrix is updated to reflect the new distances between the merged cluster and the remaining clusters. The process is repeated until we have the desired number of clusters. Similar to k-means, we have used sklearn's implementation of this algorithm. We have experimented with the combinations of cosine distance with average linkage and Euclidean distance with ward linkage. We stopped this algorithm when we had two final clusters.

3) Spectral clustering: The spectral clustering algorithm[1] is another popular unsupervised machine learning algorithm that uses the eigenvectors of a similarity matrix to cluster the data points. It is particularly useful for datasets with non-linear boundaries and can handle datasets with unequal cluster sizes or shapes. The first step is to construct a similarity matrix that captures the pairwise similarity between the data points. There are different ways to construct the similarity matrix, like using a Gaussian kernel or by computing a nearest neighbor graph. In this project we decided to use the nearest neighbor strategy. The similarity matrix is used to construct the Graph Laplacian matrix, which is a matrix that captures the relationship between the data points. The Laplacian matrix then undergoes an eigendecomposition to obtain the eigenvectors and eigenvalues. The eigenvectors corresponding to the k smallest eigenvalues are selected as the new feature vectors for the data points. The number of eigenvectors selected is the number of clusters we want to identify. Finally, the k-means algorithm is applied to the new feature vectors to group the data points into k clusters. For this project, we have used sklearn's implementation of this algorithm. We have set the number of clusters hyperparamter (k) to be 2 and 'nearest neighbors' as the method for computing the similarity matrix.

4) SWAY: SWAY[2] is an optimization algorithm that utilizes a clustering algorithm to cluster the data into 2 sets and uses an evaluation function to evaluate the two clusters. It recursively repeats this process on the best cluster until it reaches the threshold size for clusters. The clustering algorithm used by the baseline SWAY in this project is FASTMAP. FASTMAP selects a point at random. It finds the farthest point from the chosen point in the data. Let that point be A. It then finds the farthest point from A. We'll call it B. It has now obtained two points in the data that are very far from each other. The algorithm makes a line between these two points A and B and maps the rest of the points on this line. It obtains the two clusters by splitting the line in half. These two halves along with the points A and B are returned by the algorithm for evaluation. SWAY calls the evaluation function (Zitzler indicator) using the data returned by FASTMAP to obtain the "better" half. It recursively repeats the process with the "better" half until the cluster size becomes smaller than the hyperparamter for the cluster size.

5) Better: Better is the method that is used for evaluating a set of data with respect to our goals. When dealing with multiple goals, a domination predicate is required to decide if one set of performance measures is "better" than the other. Boolean domination and Zitzler multi-objective indicator[11] are two predicates that can be used for such a problem. According to Boolean domination, one thing is better than the other if it is no worse on any single goal and better on at least one goal. However, boolean domination is known to fail for for more than two objectives. Since, we have atleast three objectives in almost all of our data files, we decided to use the Zitzler indicator for this project. Zitzler's indicator favours B over A if jumping to A "loses" most. It measures the distance between the true Pareto front (i.e., the set of optimal solutions) and the obtained solution set in the objective space. The smaller the distance, the better the quality of the solution set. Further, we have made some modifications to the baseline better function. The baseline function uses only a single point from each cluster for evaluation. We felt that it would be better to assess each cluster with more than one evaluations. So, we have modified the method to accept a list of data points instead of single point and apply the Zitzler indicator on each of these points to calculate the aggregate score for the cluster.

6) PCA: PCA[10] is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets. We have used PCA to make a new clustering algorithm that can be used with SWAY. The idea was similar to how FASTMAP works. FASTMAP tries to make a line that connects two far apart points in the data and map the rest of the points on this line. We thought that it would be a better idea to use principal components that best fit the data, instead of this line.

We have used scikit-learn's implementation of PCA. In the first step, the fit() method is called to fit the PCA model to the data. This involves computing the principal components of the data and determining the amount of variance explained by each component. We decided to use the first principal component as it captures the most variance in the data. Once the PCA model is fitted, the transform() method is called to transform the data into the new feature space. This involves projecting the data onto the principal component obtained earlier. This new vector is used to sort the data, which is later split into two equal halves to obtain two clusters. Along with the two clusters, we also return a list of points which will be used for evaluating the clusters. The points are sampled randomly using the random library in Python.

7) Xpln: The baseline Xpln algorithm is a rule generation algorithm that works on the bins created using discretization. It uses an unsupervised discretization method called equal width discretization to create ranges of each attribute of equal width. The algorithm calculates the probability of each range selecting the "best" and multiplies it by its support to score each range; i.e. $b^2/(b+r)$. It does this using all the best and a random smaple of the rest. A "good" rule selects for lots of "best" and nothing much of the rest. So, the ranges are sorted using this score and rules are generated using the first range, the first 2 ranges, the first 3 ranges, etc. The performance of each rule is evaluated and the best rule is selected.

8) Xpln2: Our proposed explanation algorithm is similar to the baseline algorithm, but with two key differences. First, the baseline algorithm uses a greedy strategy to make the rules. We thought that it's possible that a better rule might not be explored using the greedy technique employed in the original. So, we decided to explore more combinations of rules, like how its done in Apriori and ECLAT algorithms. The second key change to the original algorithm is that with the increased combinations of rules, it would take a lot of time to explore all possible combinations of rules, so we have made a decision to explore only those combinations that have a score greater than equal to half of the best rule.

C. Data

Table I shows number of x and y attributes, and number of rows for each dataset.

1) Auto2: This dataset is hosted on the UCI machine learning repository. It contains information about cars such as fuel effeciency, engine size, horsepower, etc. The task is to find instances that maximize CityMPG, HighwayMPG and minimize weight and class attributes.

2) Auto93: Auto93 is a very similar to Auto2. The key differences are that it has more than 4 times the data but has less attributes. Here, we are optimizing for weight, acceleration and efficiency.

TABLE I

File	**#x**	**#y**	**#rows**
Auto2	19	4	93
Auto93	4	3	398
China	17	1	499
Nasa93dem	22	4	93
COC1000	17	5	1000
COC10000	22	3	10000
healthCloseIsses12mths0011-easy	5	3	10000
healthCloseIsses12mths0001-hard	5	3	10000
POM	10	3	10000
SSM	13	2	239360
SSN	17	2	53662

3) Nasa93dem: Nasa93 is the data collected by NASA in 1993 and consists of 63 software projects that were developed by NASA contractors between 1970 and 1987. The dataset contains a range of software projects, including real-time systems, command and control systems, scientific systems, and data processing systems. The task is to find instances that maximize Kloc and minimize Effort, Defects and the Months attributes.

4) COC1000: COC1000 is a subset of the COCOMO dataset which was created by Barry Boehm in the 1980s as part of his work on developing the COCOMO model for software cost estimation. The data contains information on 63 software projects that were created between 1966 and 1981 by various organizations. It contains details about each project's features, including its development time, number of developers, and number of lines of code. Additionally, it contains details about the project's development costs, such as the overall cost, the labor cost, and the overhead cost. Here we are optimizing for LOC, AEXP, PLEX, RISK and EFFORT.

5) COC10000: COC10000 is an even larger subset of the original COCOMO dataset. It is 10 times the size of the COC1000 dataset discussed previously. Unlike COC1000, here we are optimizing for just 3 features: lines of code, risk and effort.

6) healthCloseIssesEasy: This dataset contains information for hyperparamter optimization of the Random Forest Classifier in scikit-learn. It has features such as n_estimators, criterion, max_depth, etc. The task is to maximize ACC and PRED40 and minimize MRE.

7) healthCloseIssesHard: This dataset is very similar to healthCloseIssesEasy. It has the same kind of data (but different). It has the same number of features, rows and we are optimizing for the same set of features.

8) POM: POM is a dataset which was created using the POM3 model[2]. The model was used to explore the challenges of agile development such as balancing idle rates, completion rates and overall cost. The dataset contains features such as culture, criticality, team size, etc. The task is to minimize cost and idle times and maximize completion rates.

9) SSM: This dataset contains information about the configuration space of Trimesh, which is a library that is used to manipulate triangle meshes. It contains features such as F, smoother, colorGS,relaxParameter, V, Jacobi, line, zebraLine, cycle, alpha, beta, preSmoothing, postSmoothing. Here, we are optimizing for the number of iterations and the time to solution.

10) SSN: This dataset contains information about the configuration space of an X-264 video encoder. It contains features such as no mbtree, no_asm, no_cabac, no_scenecut, keyint, crf, scenecut, seek, ipratio. Here, we are optimizing for peak signal-to-noise ratio and energy.

D. Performance Measures

This project uses data from a variety of sources, as discussed in the previous section. For running our tests, each data file was preprocessed. Upon inspection, some of the data files had categorical data. All of the categorical features in the data files were label encoded using sklearn's LabelEncoder. Several files also had missing values. They were not handled during the preprocessing stage. They were removed from the data before it was sent to the clustering algorithms.

Several of our proposed clustering algorithms have stochastic components. So, all of the tests are run 20 times, each time with different random seeds. All of the hyperparameters of the baseline algorithms were left the same. The min_cluster size for sway is $\sqrt{n}$. We have kept the same minimum cluster size threshold for our other clustering algorithms. Further, we have made 10 evaluations to compare the two clusters generated by our proposed algorithms. For k-means, we have used the 10 data points closest to the cluster's centroid. For all the other clustering algorithms, we have randomly sampled 10 points from each cluster. The hyperparameters used for our clustering algorithms are in the Table II.

We have kept the same sampling size for "best" and "rest" data points for both of the explanation algorithms to keep things standardized. The "rest" sample size is four times the size of the "best" cluster.

SWAY and each of the SWAY replacements were run 20 times with different random seed on each datasets. The mean

TABLE II
HYPERPARAMETERS FOR CLUSTERING

Algorithm	Hyperparameters
K-Means	n_clusters=2, n_init=10
Agglomerative	n_clusters=2, metric='euclidean', linkage='ward'
Spectral	n_clusters=2, affinity='nearest_neighbors', n_jobs=10
PCA	n_components=1

of the results was calculated and used in our results. The same was done for both of the explanation algorithms.

E. Summarization Methods

Our results table contain the conjunction of an effect size test and a significance test which was created using Cliff's delta test and Bootstrapping.

Cliff's delta is a non-parametric test that measures the effect size of the difference between two groups of ordinal data. The Cliff's delta test compares the frequency with which an element in one group is greater than an element in the other. This allows us to determine how frequently one group is larger than the other. Bootstrapping uses random sampling with replacement and a delta() function that calculates the difference in means between y and z. We have also used Scott-Knott procedure for our experiments in conjuction with non-parametric effect size and significance tests.

TABLE III
SUMMARY OF CONTRIBUTIONS

Small dataset	Agglomerative clustering; ward linkage; euclidean distance
Large dataset	Map to first principal component; sort and divide
Zitzler's modification	Aggregate scores for each cluster using multiple instances
Xpln2	Explore more combination of rules with higher score threshold

IV. RESULTS

Our results aim to answer the research questions which we introduced earlier - **(RQ1)** : How does SWAY2's runtime compares to SWAY [2]?, **(RQ2)** : Can SWAY2 handle multiple minimization and maximization objectives with reasonable performance as compared to SWAY [2] and **(RQ3)**: Can XPLN2 find better combination of rules and keep explanation tax low as compared to baseline explanation algorithm, XPLN1?

Tables IV - XIV shows the results of different algorithms for each dataset. For each table, the top part shows the mean results over 20 repeated runs with different random seeds for each algorithms and the bottom part shows the conjunction of a effect size test (Cliff's Delta) and a significance test (Bootstrapping). "all" shows the raw values of all rows in the data. "sway1" show values from examples found in the leaf cluster found by SWAY [2]. "sway2" show values from the examples found by SWAY2. "xpln1" and "xpln2" shows values

TABLE IV
MEAN RESULTS FOR AUTO93

	Lbs-	Acc+	Mpg+
all	2970.42	15.57	23.84
sway1	2137.41	16.78	32.3
$*xpln1$	2336.88	15.48	27.18
$*sway2$	1832.79	17.12	34.24
xpln2	2398.5	13.25	23.86
top	1986.6	19.36	40.82
all to all	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$

TABLE V
MEAN RESULTS FOR AUTO2

	CityMPG+	HighwayMPG+	Weight-	Class-
all	22.37	29.09	3072.9	19.51
sway1	26.43	32.86	2686.66	15.5
xpln1	17.9	22.33	1924.77	11.33
$*sway2$	38.0	42.0	1976.67	8.73
$*xpln2$	30.5	34.0	2176.25	9.2
top	26.15	29.89	1586.52	6.99
all to all	=	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$	$\neq$

TABLE VI
MEAN RESULTS FOR CHINA

	N_effort-
all	4277.64
sway1	1953.71
$*xpln1$	638.16
$*sway2$	191.96
xpln2	1729.576
top	69.58
all to all	=
all to sway1	$\neq$
all to sway2	$\neq$
sway1 to sway2	$\neq$
sway1 to xpln1	$\neq$
sway2 to xpln2	$\neq$
sway1 to top	$\neq$

TABLE VII
MEAN RESULTS FOR COC1000

	LOC+	AEXP-	PLEX-	RISK-	EFFORT-
all	625.03	2.97	3.05	6.68	30807.5
sway1	976.01	2.94	3.01	5.72	26134.03
$*xpln1$	347.77	1.03	1.07	2.18	9012.26
$*sway2$	1316.49	2.83	2.9	6.23	34887.66
xpln2	994.48	2.97	3.04	7.16	30925.52
top	1013.05	0.66	0.51	1.99	14226.21
all to all	=	=	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$	$\neq$	$\neq$

TABLE VIII
MEAN RESULTS FOR COC10000

	Loc+	Risk-	Effort-
all	1009.04	6.59	30506.37
sway1	1089.33	3.04	26205.95
xpln1	907.97	6.47	27912.98
$*sway2$	1578.0	5.5	24998.21
$*xpln2$	1012.75	4.67	26638.91
top	1939.72	0.41	19702.26
all to all	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$

TABLE IX
MEAN RESULTS FOR HEALTHCLOSEISSES12MTHS0001-HARD

	MRE-	ACC+	PRED40+
all	82.32	5.15	22.1
sway1	73.39	7.64	22.19
xpln1	88.33	3.38	24.46
$*sway2$	72.03	8.17	25.05
$*xpln2$	78.1	4.93	21.2
top	65.71	13.7	27.85
all to all	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$

found by applying rules which were generated by baseline explanation algorithm and our proposed explanation algorithm respectively. ”top” sorts all rows (using the Zitzler predicate) then reports values seen in the top N rows. N is the number of rows in the leaf cluster found by SWAY algorithm. As mentioned SWAY2 is hybrid approach for sampling. We are using agglomerative clustering for smaller data with fewer parameters and PCA-based clustering for larger data with many parameter. We have used PCA-based clustering for SSN and SSM datasets and agglomerative clustering for remaining datasets. The * near the algorithm in table denotes the best performer.

As it can be seen from the results table, our proposed algorithm SWAY2 provided better results as compared to SWAY [2] for both minimizing and maximizing objectives. Also, for multiple runs with multiple random seeds sampling tax (loss seen between SWAY2 algorithm and using all) is low. It is, however, non zero as we are not looking at everything.

TABLE X
MEAN RESULTS FOR HEALTHCLOSEISSES12MTHS0011-EASY

	MRE-	ACC+	PRED40+
all	92.29	-8.53	17.79
sway1	102.92	-8.9	9.37
*$xpln1$	72.01	-5.7	30.92
*$sway2$	0.0	0.0	83.33
xpln2	72.47	-8.78	31.2
top	65.71	13.7	27.85
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠

TABLE XI
MEAN RESULTS FOR NASA93DEM

	Kloc+	Effort-	Defects-	Months-
all	94.02	624.41	3761.76	24.18
sway1	101.96	461.38	3679.71	22.91
xpln1	91.19	598.61	3655.39	23.72
*$sway2$	93.52	177.93	1320.83	10.38
*$xpln2$	78.62	370.16	3120.78	19.63
top	100.27	100.07	820.58	8.85
all to all	=	=	=	=
all to sway1	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠
sway1 to sway2	≠	≠	≠	≠
sway1 to xpln1	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠
sway1 to top	≠	≠	≠	≠

TABLE XII
MEAN RESULTS FOR SSM

	NUMBERITERATIONS-	TIMETOSOLUTION-
all	30.94	506.07
sway1	13.06	246.88
*$xpln1$	27.13	431.18
*$sway2$	5.77	92.84
xpln2	41.65	674.16
top	3.66	62.47
all to all	=	=
all to sway1	≠	≠
all to sway2	≠	≠
sway1 to sway2	≠	≠
sway1 to xpln1	≠	≠
sway2 to xpln2	≠	≠
sway1 to top	≠	≠

Sampling tax from our proposed sampling algorithm is low as compared to SWAY [2] algorithm.

Our proposed explanation algorithm gave comparable results as compared to baseline explanation algorithm. It gave better results for few datasets like healthCloseIsses12mths0001-hard as it was able to find better combinations of rules. As we can see, for multiple runs with multiple random seeds explanation tax (loss seen between SWAY2 algorithm and explanation algorithm) is low.

TABLE XIII
MEAN RESULTS FOR SSN

	PSNR-	Energy-
all	44.53	1658.0
sway1	44.92	1162.91
xpln1	44.36	1404.62
*$sway2$	39.0	767.61
*$xpln2$	39.92	1380.73
top	25.99	504.54
all to all	=	=
all to sway1	≠	≠
all to sway2	≠	≠
sway1 to sway2	≠	≠
sway1 to xpln1	≠	≠
sway2 to xpln2	≠	≠
sway1 to top	≠	≠

TABLE XIV
MEAN RESULTS FOR POM

	Cost-	Completion+	Idle-
all	369.99	0.87	0.24
sway1	329.83	0.86	0.24
*$xpln1$	226.7	0.52	0.15
*$sway2$	205.63	0.86	0.24
xpln2	347.72	0.86	0.25
top	151.09	0.96	0.0
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠

TABLE XV
MEDIAN/IQR FOR AUTO2

	Median				IQR			
	cMPG+	hMPG+	W-	C-	cMPG+	hMPG+	W-	C-
sway2	33.83	38.83	2212.50	9.60	1.03	1.63	100.5	0.2
xpln2	29.86	35.48	2312.86	10.17	5.69	4.84	499.92	7.07

TABLE XVI
MEDIAN/IQR FOR AUTO93

	Median			IQR		
	Lbs-	Acc+	Mpg+	Lbs-	Acc+	Mpg+
sway2	1837.80	17.12	34.23	195.9	0.07	1.29
xpln2	2398.50	13.25	20.00	0.00	0.00	0.00

TABLE XVII
MEDIAN/IQR FOR CHINA

	Median	IQR
	N_effort-	N_effort-
sway2	188.13	19.49
xpln2	2044.2	38.99

TABLE XVIII
MEDIAN/IQR FOR COC1000

	Median					IQR				
	L+	A-	P-	R-	E-	L+	A-	P-	R-	E-
sway2	1342.79	2.82	2.85	5.45	34682.03	158.78	0.18	0.16	4.32	23265.2
xpln2	989.94	2.96	3.01	10.00	35395.17	2.65	0.002	0.01	3.45	649.52

TABLE XIX
MEDIAN/IQR FOR COC10000

	Median			IQR		
	LOC+	RISK-	EFFORT-	LOC+	RISK-	EFFORT-
sway2	1471.77	6.38	26888.07	234.52	1.81	15139.41
xpln2	1013.67	7.52	27763.76	6.34	2.54	844.13

TABLE XX
MEDIAN/IQR FOR HEALTHCLOSEISSES12MTHS0001-HARD

	Median			IQR		
	MRE-	ACC+	PRED40+	MRE-	ACC+	PRED40+
sway2	72.46	8.02	25.00	0.53	0.27	0.34
xpln2	81.93	5.25	22.70	0.92	0.21	0.78

TABLE XXI
MEDIAN/IQR FOR HEALTHCLOSEISSES12MTHS0011-EASY

	Median			IQR		
	MRE-	ACC+	PRED40+	MRE-	ACC+	PRED40+
sway2	0.00	0.00	83.33	0.0	0.0	0.0
xpln2	89.47	-8.70	20.35	0.0	0.0	0.0

TABLE XXII
MEDIAN/IQR FOR NASA93DEM

	Median				IQR			
	K+	E-	D-	M-	K+	E+	D-	M-
sway2	94.26	165.73	1122.76	9.60	1.03	1.63	100.5	0.2
xpln2	69.86	350.48	3312.86	15.17	5.69	4.84	499.92	7.07

TABLE XXIII
MEDIAN/IQR FOR SSM

	Median		IQR	
	NI-	TS-	NI-	TS-
sway2	5.71	90.88	1.33	16.75
xpln2	39.87	664.83	34.40	539.25

Tables XV - XXV report the median (50th percentile) and IQR (75th - 25th percentile) over 20 runs with different random seeds. On those populations, we have also run the statistical analysis described in III-E.

The table XXVI contains the win/loss summary of all the algorithms that we have tested throughout the experiments.

TABLE XXIV
MEDIAN/IQR FOR SSN

	Median		IQR	
	PSNR-	Energy-	PSNR-	Energy-
sway2	38.62	651.47	2.61	517.59
xpln2	44.44	1426.64	2.40	1289.6

TABLE XXV
MEDIAN/IQR FOR POM

	Median			IQR		
	Cost-	Completion+	Idle-	Cost-	Completion+	Idle-
sway2	227.23	0.87	0.25	102.51	0.02	0.05
xpln2	347.72	0.86	0.25	0.0	0.0	0.0

TABLE XXVI
RESULTS SUMMARY

Algorithm	**Wins**	**Loses**
Sway1	0	11
Sway2	11	0
Xpln1	6	5
Xpln2	5	6

V. FUTURE WORK

There is a lot of scope for experiments in this project. A few things that we couldn't do was try different sampling techniques to sample good representations from the clusters. For example, after k-means clustering, we sampled points near the center of the cluster but it might have been better to sample points closer to the boundaries as there is a better chance that they are closer to the pareto frontier. We also couldn't try more statistical methods on the results to obtain even more information on the performance of our new algorithms. Also, our final results might be affected by biases like Sampling bias and Parameter bias. This should be explored in future work.

VI. CONCLUSION

SWAY2 is comparable with SWAY at speed and generally performs better. The proposed hybrid algorithm takes advantage of the strengths of the individual algorithms that it constitutes. One performs better but takes more time and memory, so it is suitable for smaller datasets, the other, although performs slightly worse but is much faster and suitable for larger data. Our new explanation algorithm fought admirably against the baseline, but fell short, winning on 5 out of 11 datasets used in this project.

Since, the new explanation algorithm takes more time than the baseline, it's safe to conclude that the combination of our SWAY2 and the baseline XPLN would perform the best for building a multi-objective semi-supervised explanation system.

References

[1] Stephen T Barnard, Alex Pothen, and Horst D Simon. “A spectral algorithm for envelope reduction of sparse matrices”. In: *Proceedings of the 1993 ACM/IEEE Conference on Supercomputing*. 1993, pp. 493–502.

[2] Jianfeng Chen et al. ““Sampling” as a baseline optimizer for search-based software engineering”. In: *IEEE Transactions on Software Engineering* 45.6 (2018), pp. 597–614.

[3] Kalyanmoy Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[4] John A Hartigan and Manchek A Wong. “Algorithm AS 136: A k-means clustering algorithm”. In: *Journal of the royal statistical society. series c (applied statistics)* 28.1 (1979), pp. 100–108.

[5] Joseph Krall, Tim Menzies, and Misty Davies. “Gale: Geometric active learning for search-based software engineering”. In: *IEEE Transactions on Software Engineering* 41.10 (2015), pp. 1001–1018.

[6] Tansel Özyer, Reda Alhajj, and Ken Barker. “Clustering by integrating multi-objective optimization with weighted k-means and validity analysis”. In: *Intelligent Data Engineering and Automated Learning–IDEAL 2006: 7th International Conference, Burgos, Spain, September 20-23, 2006. Proceedings 7*. Springer. 2006, pp. 454–463.

[7] Ruhul Sarker, Ko-Hsin Liang, and Charles Newton. “A new multiobjective evolutionary algorithm”. In: *European Journal of Operational Research* 140.1 (2002), pp. 12–23.

[8] Nidamarthi Srinivas and Kalyanmoy Deb. “Muiltiobjective optimization using nondominated sorting in genetic algorithms”. In: *Evolutionary computation* 2.3 (1994), pp. 221–248.

[9] Joe H Ward Jr. “Hierarchical grouping to optimize an objective function”. In: *Journal of the American statistical association* 58.301 (1963), pp. 236–244.

[10] Svante Wold, Kim Esbensen, and Paul Geladi. “Principal component analysis”. In: *Chemometrics and intelligent laboratory systems* 2.1-3 (1987), pp. 37–52.

[11] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. “SPEA2: Improving the strength Pareto evolutionary algorithm”. In: *TIK-report* 103 (2001).