CSC 591 Project: A Better Sway?

Anant Gadodia
College of Engineering
North Carolina State University
Raleigh, NC 27695
Email: agadodi@ncsu.edu
Krish Jain
College of Engineering
North Carolina State University
Raleigh, NC 27695
Email: kdjain@ncsu.edu
Manasi Ghosalkar
College of Engineering
North Carolina State University
Raleigh, NC 27695
Email: mghosal@ncsu.edu

***Abstract*—In today's times of advanced Artificial Intelligence helping developers code and codes becoming more complex, Testing of the code to ensure quality has become more important than ever. In this program, we look at TESTED by Dr. Tim Menzies to understand how a black-box optimization tool helps us optimize our codes and understand the code for all stakeholders. In this project we will be attempting to improve the existing tool by re-configuring certain aspects of the code to see the affect on scalability, accuracy and run times.**

I. Introduction

Clustering is a common technique used in machine learning and data mining to group similar objects together based on a set of criteria. Data mining and machine learning both frequently utilize the clustering technique to group related objects based on a set of criteria. To overcome some of the drawbacks of conventional clustering techniques, two strategies have been developed: semi-supervised clustering and multi-objective clustering. These techniques do have some drawbacks, though, including sensitivity to initialization, difficulties figuring out the ideal number of clusters, and the requirement for in-depth domain knowledge. Researchers have created semi-supervised clustering and multi-objective clustering techniques to overcome these constraints.[1]
Multi-objective clustering, on the other hand, aims to optimize multiple clustering objectives simultaneously, such as maximizing intra-cluster similarity and minimizing inter-cluster similarity. However, these methods also have their own set of challenges, such as determining the appropriate trade-off between different objectives and dealing with a large number of conflicting objectives.[2]
Another strategy for enhancing the system's performance is search-based software engineering (SBSE). By considering the software development process as an optimization problem, SBSE automates it using search-based optimization approaches. SWAY is a method used in SBSE that intelligently chooses a subset of the search space to investigate, speeding up the search process.[3]
In this report, we will provide an overview of multi-objective clustering, explain and implement certain modifications in the Sway algorithm specifically by implementing Jaccard Distance Similarity and adding k-means methods to further improve the clustering accuracy of the algorithm and discuss the challenges associated with these methods.

II. Related Work

Jaccard distance is a similarity measure commonly used in clustering to measure the similarity between sets of objects. It is based on the idea of calculating the ratio of the intersection of two sets to the union of those sets. In a multi-object clustering method, Jaccard distance can be used as a similarity measure to cluster objects based on their similarity with respect to a set of attributes. For example, optimizing the SWAY algorithm to calculate this distance rather than calculating the Cosine similarity distance based on a threshold can be used to cluster NUMs and SYMs based on their similarity with respect to the input features.[5]
KMeans is a clustering algorithm that partitions a dataset into a predetermined number of clusters based on their proximity to a set of centroids. KMeans can be used in SWAY to cluster objects based on their similarity with respect to a set of attributes. However, it is very sensitive to initial starting points. Hence, using different techniques to pre-calculate the initial centroids can in turn result in better solutions.[7]
Zitzler's continuous domination predicate (CDP) is a mathematical framework that can be used to optimize multiple objectives simultaneously, based on the idea of ranking solutions based on their ability to dominate other solutions in the search space. Boolean Domination (BDOM) can be used to optimize multiple objectives, such as maximizing the accuracy of the clustering results while minimizing the number of clusters.[4]
In summary, Jaccard distance, KMeans, and Zitzler's CDP are all techniques that can be used in Sway to improve the clustering results and optimize multiple objectives simultaneously. By combining these techniques with multi-objective clustering and semi-supervised approaches, researchers can create more effective and efficient Rules that provide accurate and interpretable explanations for their decisions.[6]

III. Methods

A. Dataset

1) Auto2: Auto2.csv [22] is a Car Mileage dataset containing 23 attributes and 93 observations. The dataset contains multiple categorical attributes related to Car type, make along with all car-related performance metrics to effectively calculate mpg, weight, and class.

2) Auto93: Auto93.csv [22] is a Car Mileage (mpg) Multivariate dataset containing 8 Attributes and 398 samples to predict the mpg attribute based on other attributes. It is concerned with city-cycle fuel consumption in miles per gallon and comprises 3 multivalued discrete and 5 continuous attributes. This dataset is associated with a regression task and can also be used for clustering purposes

3) China: China.csv [23] is a dataset expressed in COCOMO format which is used for predicting Software effort estimation (SEE). This dataset is used as an experiment to learn good predictors and emphasis is put more on choice of data to be collected rather than what learner is applied.

4) Coc1000: Coc1000.csv[23] is a dataset used to evaluate new estimation methods based on Software effort estimation. This data contains 1000 observations and 25 attributes, based on the COCOMO format for methods ranging from classical to modern feature selection, which can be used to run comparison experiments on.

5) Coc10000: Coc1000.csv[23] is a dataset with similar attributes to that of the coc1000 dataset but in turn, it has 10,000 observations which can be used as a sufficient sample size for testing out the scalability of Software effort estimation methods.

6) HealthCloseIsses12mths0001-hard:
HealthCloseIsses12mths0001-hard.csv[24] contains 8 atributes and 10,000 observations can be used for hyper-parameter optimzations of Random Forest classifiers. This dataset captures the time related to health diseases and can be potentially used to analyze the efficiency

and effectiveness of the health related issues. This will be an ideal dataset to check for various statistical analysis to extract insights

7) HealthCloseIsses12mths0001-easy: HealthCloseIsses12mths0001-easy.csv[24] is similar to HealthCloseIsses12mths0001-hard dataset with 10,000 observations and 8 attributes. Classifiers trained of hard dataset can be used to verify on this particular dataset. Hyper-parameters can be optimized on one dataset and verified on other dataset.

8) Nasa93dem: Nasa93dem.csv[23] file containes 93 observations and 29 attributes being used for Software effort estimation and estimation detection as it captures the effort and time required to maintain projects at NASA. All the attributes of this dataset can be projected onto a cluster and can be also checked for any information loss during the projection. Because of close inter-cluster contrast sets we can find very small rules in logLinear time. The goal of this dataset is to improve project planning and create reliable software effort models

9) Pom: Pom.csv[25] contains 13 attributes and 10,000 observations which can be used to analyze and optimize Agile project management practices for software development tasks, which is essential for successful project delivery. The goal of this dataset is to enable developing models and techniques for optimizing Agile project management practices like improving completion rates, reducing idle rates, and reducing overall cost.

10) SSM: SSM.csv[26] contains information related to the configuration space of Trimesh, a library used to manipulate triangle meshes. It provides information about the solver Trimesh, including the number of iterations and the various smoothing techniques used to solve the mesh-related problems. These techniques include F, smoother, colorGS, relaxParameter, V, Jacobi, line, zebraLine, cycle, alpha, beta, preSmoothing, and postSmoothing. This dataset can be used to optimize and improve the performance of Trimesh and better understand the complexities of the configuration space.

11) SSN: SSN.csv[26] contains attributes related to video coding and compression. Each attribute represents a different setting which can be adjusted to affect the quality and efficiency of video encoding. For example, ”no mbtree” indicates whether or not macroblock tree coding is used, while ”Qp” represents the quantization parameter used for encoding. Other attributes include ”Ref” (the number of reference frames used), ”Bframes” (the number of B frames used), and ”Keyint” (the maximum interval between key frames). This dataset can be used to identify patterns between these attributes and the resulting video quality and energy consumption.

B. Algorithms

1) Sway: Sway is a multi-objective semi-supervised algorithm that returns a cluster with the best performance over a set of target objectives. Starting with a random point A, choose a point B that is distant to A. Project the other points onto a line connecting A and B and depending on their distance from A and B, divide the data into two halves. The distance between two points considered here is cosine distance using random projections. After separating the points into two halves, choose the better half with respect to the objective function or in this case, target variables and apply the same process recursively on that half. For computing which half is better, Zitzler's continuous domination predicate is used. For every recursive call we carry over the border point A or B depending on the chosen half to the next call as A. This process is repeated until the size of the chosen half becomes less than or equal to a predetermined fraction of the size of the original dataset. The function thus gives a cluster that represents the 'best' subset of the data with respect to the target y variables. It also outputs 'rest' that takes random examples other than the ones in 'best'.

To get random projections and then separate the data points, we use local sensitivity hashing. Here, a random subset of data is chosen, projections of those points are taken with respect to the line joining A and B and the distance of these projections from A and B is used for comparison. On a base level, L2-Norm is used for distance between two rows of data.

Zitzler's predicate [4] is a continuous domination predicate that judges which row dominates among a pair of rows. It does so by evaluating the change of status while jumping from one point to another. We compute forward jump s_1 and backward jump s_2 as:

$$s_1 = -\sum_i (e^{w_i * (a_i - b_i)/n})$$

$$s_2 = -\sum_i (e^{w_i * (b_i - a_i)/n})$$

where a and b refer to the two rows of data, a_i and b_i refer to the i^{th} column values, and w_i is the weight associated with the goal i. If $s_1 < s_2$, we can say that we lost the least jumping to a_1 and therefore, a_1 is the superior combination.

2) Sway2 - Jaccard Distance: Given two rows of data, to capture the difference in the data, we propose to use a modified Jaccard[5] distance as opposed to the standard L2-Norm or Euclidean distance. Given two rows row1 and row2, compare the column values and assign $d_i = 0$ or 1 depending on whether the i^{th} column of row1 and row2 matches or not. Generally, $d_i = 0$ if $x == y$ else 1 for both symbols and numbers. For continuous valued variables, $d_i = 0$ if the normalized absolute distance between x and y is less than a predetermined threshold, otherwise, $d_i = 1$. Finally, the distance is calculated as $(\sum_i d_i)/(n + \sum_i d_i)$. This is likely to reduce computational effort moving from the L2-Norm distance approach.

3) Sway3 - KMeans: Kmeans[21] is a clustering algorithm designed for dividing the data into clusters of close enough data points. It is an iterative algorithm that starts with k random points. Label all points according to their closest k. Then reassign k to the mean value of the points labeled with the corresponding k. Repeat until the difference in the k positions over consecutive iterations becomes less than a threshold value.

Since K-Means is an unsupervised clustering algorithm, to apply it to this particular semi-supervised setting, we modified $sway1$ by using K-Means to split the data into two halves. Then, choosing the better half according to Zitzler's predicate applied on the target variables, apply the same process recursively until the size of the cluster is less than or equal to a preset fraction of the size of the dataset. The algorithm thus returns the 'best' cluster it can find.

4) Xpln1: An explanation system is used to describe the $best$ cluster output as given by the sway function by establishing 'rules' and see how it compares to the other data points, i.e., $rest$. After obtaining $best$ and $rest$ from sway, binning is used to find ranges among the feature space for individual features such that it can provide a reasonable classification point for that variable. Starting with 16 initial buckets, we recursively merge adjacent ranges with similar distributions by seeing how many $best$ and $rest$ samples can be found within those ranges. Eventually, we get ranges that can provide a better basis for a somewhat naive classification.

Then, we explore these ranges to get $rules$ that point to the best cluster. Sort the ranges based on their ability to select 'best' i.e. $b^2/(b + r)$ where b and r are the number of best and rest samples found in a range. Create rules using these ranges in order such that r1 considers the top ranked range, r2 considers the first 2 ranges, r3 considers the first 3 ranges and so on. Select the best rule using the same formula used earlier to sort the ranges. The rule provided by the xpln algorithm provides constraints and ranges that can be used to select data points yielding good target values in a simpler and explainable way.

5) Xpln2 - Decision Trees: We took the best and rest outputs obtained from the sway function to train a decision tree classifier[19][20]. Using the outputs from the decision tree classifier, we see how it is able to select the data-points with the optimum target values. This algorithm works by using the best and rest provided by using sway on the sampled data and then fitting this data to a Decision Tree Classifier. This classifier is then used to predict whether additional data points will be optimal(Best) or not(Rest). This will help us apply sway to a widely used simpler algorithm which increases understandability for all stakeholders.

C. Performance Measures

The performance of the above mentioned methods was compared using boolean domination over the target variables. We run sway1, sway1 with xpln1, sway2 (Jaccard) with xpln2 and sway3 (K-Means) with xpln2. We run these combinations for all data sets 20 times and generate corresponding outputs. We then compare the results for the corresponding runs using simple Boolean Domination. For every target variable, we see which of the methods outperform the other. We also perform running time analysis on these iterations to evaluate which methods are more time-efficient.

D. Summarization Methods

We use the Kruskal-Wallis [17] test to compare the change in performance of the methods over the baseline sway method. If the p-value obtained by comparing outputs for method 1 and method 2 is below the significance level (0.05), the null hypothesis is rejected and we can say that there is a difference in outputs of methods 1 and 2.

IV. Results

In this paper we have worked on improving the Multi-Objective Semi-Supervised Clustering and explanation system Sway and XPLN . In our method, we have replaced the Eucleadian and Cosine distance with Jaccard distance and in doing this we noticed a improvement in the run time and better results. We also tried implementing K-Means clustering to the sway algorithm and the results are as below.

	P Value with KW Test with Sway 1	
Dataset	**Jaccard Improvement**	**Jaccard and K-Means Improvement**
Auto2	0.052	0.260
Coc1000	0.041	0.16
Coc10000	6.57e-5	0.017
Auto93	0.0343	0.652

TABLE I
P Value for Various Data Sets

Based on the Table I below we can conclude that the p values obtained without K-Means are better. We believe that this is because the clustering method used by sway is better than K-Means. We also noticed that K-Means was much slower as compared to the original method.

Data set	Data Size	Avg Runtime of 20 Iterations		% Change over Sway
		Sway 1	Jaccard and K-Means Improvement	
Auto2	93x23	0.0595	0.0684	14%
Auto93	398x8	0.0218	0.1141	423%
Coc1000	1,000x25	0.2841	0.1879	-33%
Coc10000	10,000x25	1.1961	2.3022	92%

TABLE II
Run time Analysis of Sway and K-Means

Table II shows a comparison of the run times of both the algorithms for various data sets and we can see that in most cases K-Means is much slower than Sway especially when we are trying to scale the data set.
We then compare the values generated for each target variable for Boolean Domination. For comparison with K-Means, we have used the Auto93 data set and the values generated for the target columns by each algorithm can be found in table III

Algorithm	Column Name		
	Lbs-	**MPG+**	**Acc+**
All	2970.41	23.84	15.57
Sway1	2288.25	26.67	15.71
Sway2(Jaccard)	2147.77**	31.54**	17.1**
Sway3(Jaccard + K-Means)	2492.3	26	15.5
Xpln1	2423.3	28.57	16.79
Xpln2(Jaccard)	2179.57	28.57	16.61
Xpln3(Jaccard + K-Means)	2511.86	27.59	15.18

TABLE III
Column Values for Auto93

Based on the above table we can conclude that the best value by Boolean domination is for Sway 2 with it dominating each variable. Hence we decided to eliminate K-Means moving forward and apply only the Jaccard distance metric.
We will now compare the results for Sway 1 and Sway 2 for the 4 data sets mentioned in table I

A. Auto2

Auto2 has 93 examples for 23 columns including 4 target variables. The values for the target variables obtained can be seen in table IV. Based on this data we can conclude that by means of binary domination, the algorithm Sway 2 produces better results.
Figures 6 to 9 in the appendix show the various values obtained for each variable in the algorithm

Algorithm	Columns			
	CityMPG+	**Class-**	**HighwayMPG+**	**Weight**
All	22.37	19.51	29.09	3072.9
Sway1	31.5	10.8	37	2286.67
Sway2(Jaccard)	34.8**	8.38**	39.2**	1958**
Xpln1	29.56	10.17	35.48	2312.86
Xpln2(Jaccard and Decision Tree Classifier)	34.8	8.38	39.2	1958
Top	20.8	29.32	27.6	3337

TABLE IV
Column Values for Auto2

Algorithm	Columns			
	CityMPG+	**Class-**	**HighwayMPG+**	**Weight**
All to All	=	=	=	=
All to Sway 1	≠	≠	≠	≠
All to Sway 2	≠	≠	≠	≠
Sway 1 to Sway 2	≠	≠	≠	≠
Sway 1 to Xpln 1	≠	≠	≠	≠
Sway 2 to Xpln 2	=	=	=	=

TABLE V
Column Value Comparison for Auto2

Table VI shows the comparison of the average run times for both the sway algorithms and Figure 1 shows the run times for each iteration of the code.
Based on this we can conclude that for this data set, the Run time of Sway2 is much faster than Sway1.
The P values obtained are 0.0524 for Sways and 0.0432 for Xplns and hence we can conclude that the data obtained is significantly different.

Algorithm	AVG Run time
Sway1	0.058
Sway2(Jaccard)	0.277

TABLE VI
RUN TIMES FOR AUTO2

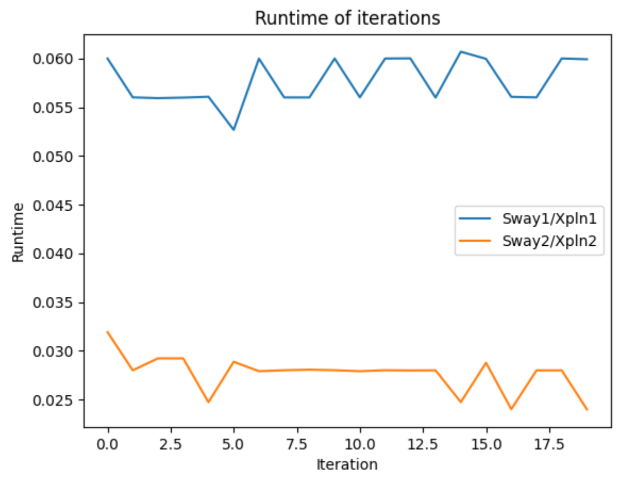

Fig. 1. Run times for Auto2

B. *Auto93*

Auto93 has 398 examples for 8 columns including 3 target variables.
The values for the target variables obtained can be seen in table VII. Based on this data we can conclude that by means of binary domination, the algorithm Sway 2 produces better results.
Figures **??** to 12 in the appendix show the various values obtained for each variable in the algorithm

Algorithm	Columns		
	ACC+	LBS-	MPG+
All	15.57	2970.42	23.84
Sway1	15.71	2288.25	26.67
Sway2(Jaccard)	17.1*	2147.77*	31.54*
Xpln1	16.79	2423.3	28.57
Xpln2(Jaccard and Decision Tree Classifier)	16.61	2179.57	28.57
Top	13.04	4624.46	10

TABLE VII
COLUMN VALUES FOR AUTO93

Algorithm	Columns		
	ACC+	LBS-	MPG+
All to All	=	=	=
All to Sway 1	$\neq$	$\neq$	$\neq$
All to Sway 2	$\neq$	$\neq$	$\neq$
Sway 1 to Sway 2	$\neq$	$\neq$	$\neq$
Sway 1 to Xpln 1	$\neq$	$\neq$	$\neq$
Sway 2 to Xpln 2	$\neq$	$\neq$	$\neq$

TABLE VIII
COLUMN VALUE COMPARISON FOR AUTO93

Table IX shows the comparison of the average run times for both the sway algorithms and Figure 2 shows the run times for each iteration of the code.
Based on this we can conclude that for this data set, the Run time of Sway2 is faster than Sway1.

Algorithm	AVG Runtime
Sway1	0.361
Sway2(Jaccard)	0.0311

TABLE IX
RUNTIMES FOR AUTO93

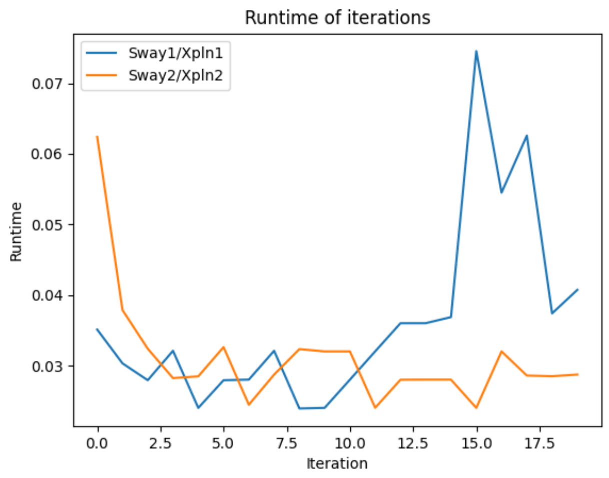

Fig. 2. Run times for Auto93

The P values obtained are 0.0.34 for Sways and 0.985 for Xplns and hence we can conclude that the data obtained is significantly different for Sways.

C. *Coc1000*

COC1000 has 1000 examples for 25 columns including 5 target variables.
The values for the target variables obtained can be seen in table X. Based on this data we conclude that by Boolean Domination, the results generated by Sway 1 and Sway 2 are similar.
Figures 13 to 17 in the appendix show the various values obtained for each variable in the algorithm

Algorithm	Columns				
	AEXP-	Effort-	Loc+	Plex-	Risk-
All	2.97	30807.5	1013.05	3.05	6.68
Sway1	3.1	19590.77	1138.29*	2.94*	4.71*
Sway2(Jaccard)	3.13	14939.57*	789.65	2.94*	4.71*
Xpln1	2.99	26293.59	980.1	3.02	6.34
Xpln2(Jaccard and Decision Tree Classifier)	2.98*	24907.03	950.8	3.03	6.34
Top	2.74	29676.65	932.9	3.29	5.94

TABLE X
COLUMN VALUES FOR COC1000

Table XII shows the comparison of the average run times for both the sway algorithms and Figure 3 shows the run times for each iteration of the code.
Based on this we can conclude that for this data set, the Run time of Sway2 is much faster than Sway1.

The P values obtained are 0.0405 for Sways and 0.589 for Xplns and hence we can conclude that the data obtained is significantly different for Sways.

Algorithm	Columns				
	AEXP-	Effort-	Loc+	Plex-	Risk-
All to All	=	=	=	=	=
All to Sway 1	≠	≠	≠	≠	≠
All to Sway 2	≠	≠	≠	≠	≠
Sway 1 to Sway 2	≠	≠	≠	=	=
Sway 1 to Xpln 1	≠	≠	≠	≠	≠
Sway 2 to Xpln 2	≠	≠	≠	≠	≠

TABLE XI
COLUMN VALUE COMPARISON FOR COC1000

Algorithm	AVG Runtime
Sway1	0.1469
Sway2(Jaccard)	0.0867

TABLE XII
RUNTIMES FOR COC1000

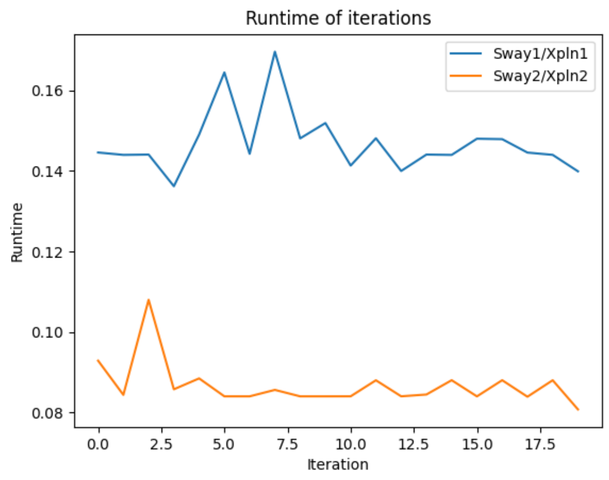

Fig. 3. Run times for COC1000

D. Coc10000

COC10000 has 10,000 examples for 25 columns including 3 target variables.
The values for the target variables obtained can be seen in table XIII. Based on this data we conclude that by Boolean Domination, the results generated by Sway 1 are better than Sway 2.
Figures 18 to 20 in the appendix show the various values obtained for each variable in the algorithm

Table XV shows the comparison of the average run times for both the sway algorithms and Figure 4 shows the run times for each iteration of the code.
Based on this we can conclude that for this data set, the Run time of Sway2 is faster than Sway1. from the figure 4 we observe that though initially sway 1 is faster than sway2, as the number of iterations increases, the run time for sway 1 sees a drastic increase. to test this theory, we reran the algorithm with 50 iterations and found out that in such a scenario, sway2 performs much better than sway 1. the average run time can be seen in table XVI and the run times for each iteration can be seen in Figure 5

Algorithm	Columns		
	Loc+	Rick-	Effort-
All	1009.04	6.59	30506.37
Sway1	1031.41	3.01*	12629.55*
Sway2(Jaccard)	1073.68*	4.32	23439.58
Xpln1	1009	6.75	33580.05
Xpln2(Jaccard and Decision Tree Classifier)	1035.59	5.56	26133.67
Top	1026.6	6.74	33689.19

TABLE XIII
COLUMN VALUES FOR COC10000

Algorithm	Columns		
	Loc+	Rick-	Effort-
All to All	=	=	=
All to Sway 1	≠	≠	≠
All to Sway 2	≠	≠	≠
Sway 1 to Sway 2	≠	≠	≠
Sway 1 to Xpln 1	≠	≠	≠
Sway 2 to Xpln 2	≠	≠	≠

TABLE XIV
COLUMN VALUE COMPARISON FOR COC10000

Algorithm	AVG Runtime
Sway1	2.0702
Sway2(Jaccard)	1.9881

TABLE XV
RUNTIMES FOR COC10000

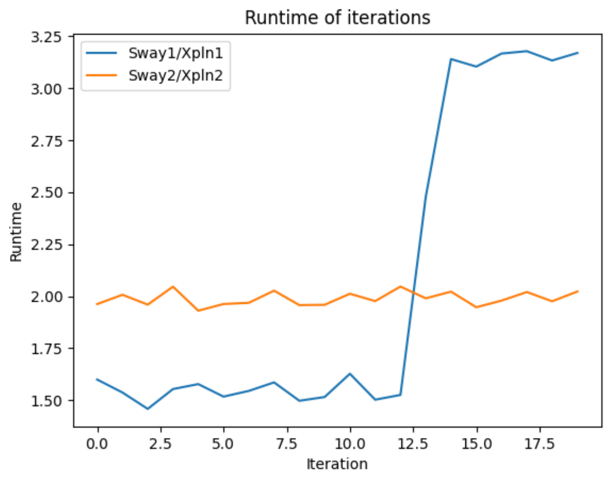

Fig. 4. Run times for COC1000

Algorithm	AVG Runtime
Sway1	2.0702
Sway2(Jaccard)	1.208

TABLE XVI
RUNTIMES FOR COC10000 50 ITERATIONS

The P values obtained are $6.578e^{-5}$ for Sways and $1.83e^{-7}$ for Xplns and hence we can conclude that the data obtained is significantly different for both.

V. DISCUSSION

In the results we have seen our algorithm compare to the state of the art for multiple times and we try to determine the best algorithm

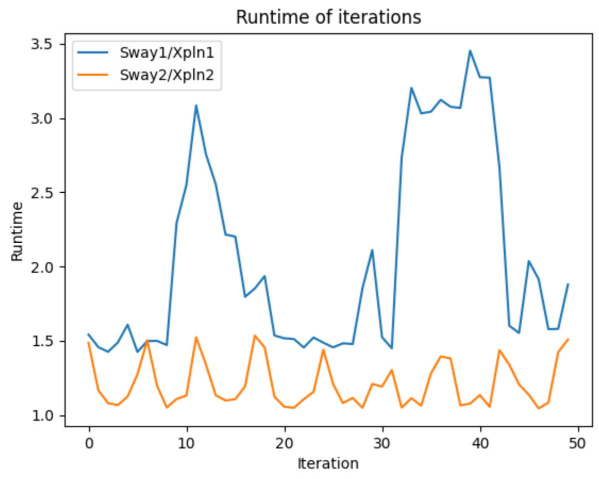

Fig. 5. Run times for COC1000 iwth 50 iterations

based on the parameters Run time and Boolean Domination. We see that in most cases, our algorithm has performed better in both the parameters. we believe that the Jaccard distance reduces the complexity of the calculation while maintaining accuracy and hence we propose that the Jaccard distance be used to replace the Euclidean distance metric being used in the state of the art algorithm. We also observe that in cases with fewer iterations, the impact of the change is not as high as that in cases with higher iterations. We also observe that the scalability of the system is good as the impact increases with an increase in the scale of the efficiency of the system increases. One threat we notice is the ability to handle various types of data sets and this is something we hope to study in the near future and understand the impact of this algorithm.

VI. Conclusion

Based on the above data obtained by running the improved algorithm for various data sets, we can conclude that the method with the Jaccard distance is in fact a improvement to the system. We would like to further study the impact on a larger Set of data sets to understand the adaptability of this algorithm on various types of data sets. We also recommend trying out other hyper parameter tuning to further improve this algorithm.

References

[1] H. Zhou, C. Li, F. Zhu, and Z. Zhang, *"A Survey of Clustering Techniques in Machine Learning* in IEEE Access, vol. 8, pp. 55100-55118.

[2] N. H. Nguyen, M. Ester, and J. Bailey, *M*ulti-Objective Clustering: Recent Developments and Future Directions in IEEE Transactions on Knowledge and Data Engineering, vol. 29, no. 11, pp. 2434-2450, 1 Nov. 2017

[3] M. Harman, P. McMinn, and J. Wegener, *S*earch-Based Software Engineering: Trends, Techniques and Applications, in ACM Computing Surveys, vol. 45, no. 1, pp. 1-61, 2012,

[4] E. Zitzler and S. Kunzli *Indicator-BasedSelectionin MultiobjectiveSearch* https://www.simonkuenzli.ch/docs/ZK04.pdf

[5] Huang, H., Zhang, Y., & Wu, W. (2017). *Multi-objective clustering with Jaccard distance* Journal of Intelligent and Fuzzy Systems, 33(1), 383-390

[6] Zhao, J., Feng, Z., & Wang, L. (2019). *A multi-objective clustering algorithm based on Jaccard distance.* Journal of Intelligent and Fuzzy Systems, 37(3), 3373-3383.

[7] Zhang, S., Zheng, S., & Gao, X. (2017). *Multi-objective clustering algorithm based on KMeans and improved NSGA-II* Journal of Intelligent and Fuzzy Systems, 32(5), 3241-3251.

[8] Tavakoli, A., & Minaei-Bidgoli, B. (2018). *Multi-objective KMeans algorithm for text clustering* Journal of Intelligent and Fuzzy Systems, 34(4), 2561-2568.

[9] Li, C., Li, X., Li, J., & Zhang, B. (2021). *A multi-objective KMeans clustering algorithm based on evolutionary algorithm for big data* Cluster Computing, 24, 4375-4387.

[10] Zhao, X., Cheng, J., Zhang, Y., & Yang, Y. (2021). *A multi-objective clustering algorithm based on Itzler's continuous domination predicate* Journal of Ambient Intelligence and Humanized Computing, 12, 6343-6354.

[11] Guo, S., Zhao, J., & Zhang, Q. (2021). *A multi-objective clustering algorithm based on Itzler's continuous domination predicate and KMeans* Applied Sciences, 11, 11192.

[12] Azzini, A., Innocenti, E., Vanneschi, L., & Tonda, A. (2021). *Multi-objective clustering with random projection and Pareto dominance* Applied Soft Computing, 107, 107406.

[13] Mohamad, M. S., & Ting, I. H. (2017). *Multi-objective clustering using random projections and NSGA-II* In 2017 IEEE Congress on Evolutionary Computation (CEC) (pp. 1755-1762). IEEE.

[14] Ma, L., Liu, Y., Xu, X., & Yao, J. (2020). *Multi-objective semi-supervised clustering based on improved NSGA-II* Applied Soft Computing, 93, 106337.

[15] Zou, Y., Chen, H., Li, Z., Wang, X., & Lu, C. (2020). *Semi-supervised multi-objective clustering based on KMeans and self-training* Information Sciences, 528, 84-101.

[16] Kavitha, P. R., & Raja, S. (2019). *Semi-supervised multi-objective clustering for gene expression data* Cluster Computing, 22, 2617-2628.

[17] W. H. Kruskal & W. W. Wallis, *Use of Ranks in One-Criterion Variance Analysis* Journal of the American Statistical Association, Vol. 47, Issue 260, pp. 583-621, 1952.

[18] Quinlan,R. (1993). *Combining Instance-Based and Model-Based Learning* In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst. Morgan Kaufmann.

[19] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees* Wadsworth, Belmont, CA, 1984.

[20] https://scikit-learn.org/stable/modules/tree.html

[21] https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html

[22] https://archive.ics.uci.edu/ml/datasets/auto+mpg

[23] https://arxiv.org/pdf/1609.05563.pdf#page=5

[24] https://scikit-https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[25] https://arxiv.org/pdf/1608.07617.pdf

[26] https://arxiv.org/pdf/1801.02175.pdf#page=2

VII. Appendix

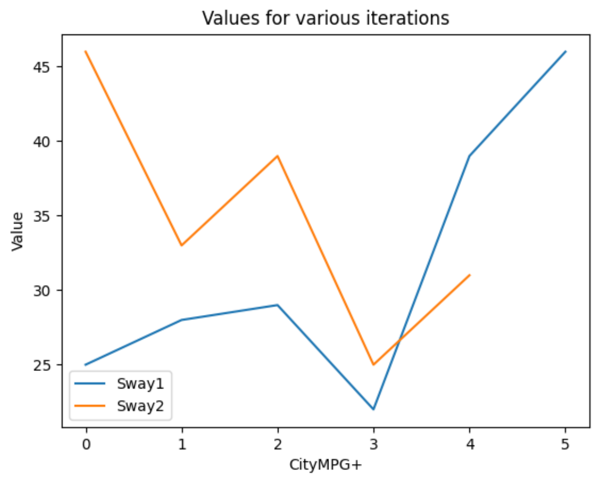

Fig. 6. Values for City MPG: Auto 2

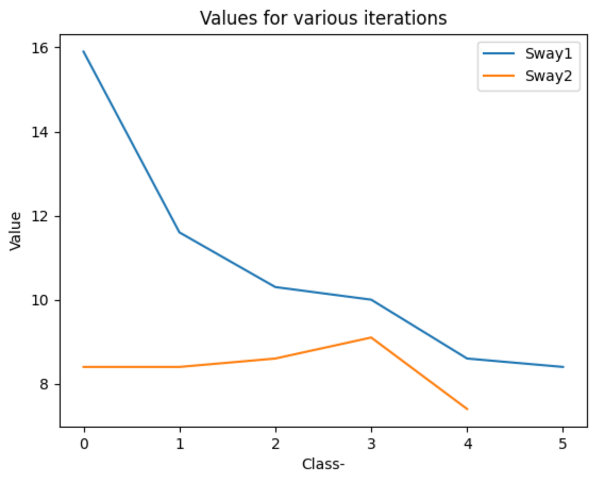

Fig. 7. Values for Class: Auto 2

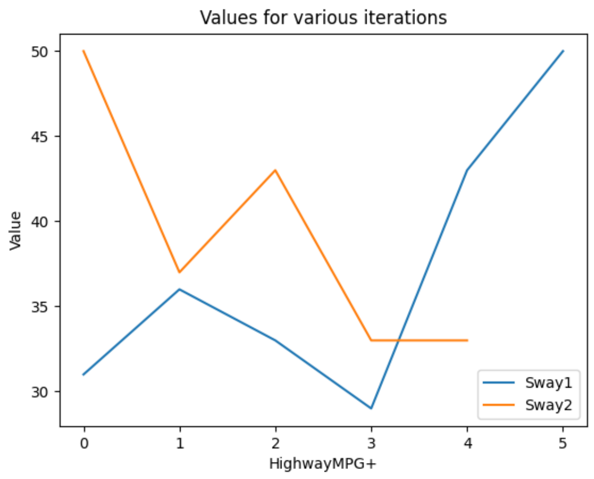

Fig. 8. Values for Highway MPG: Auto 2

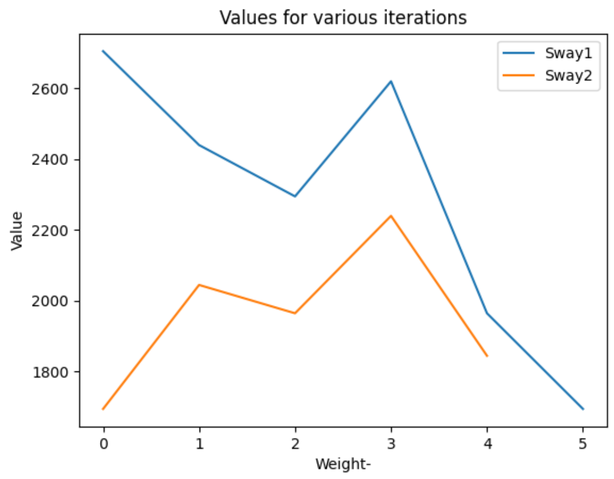

Fig. 9. Values for Weight: Auto 2

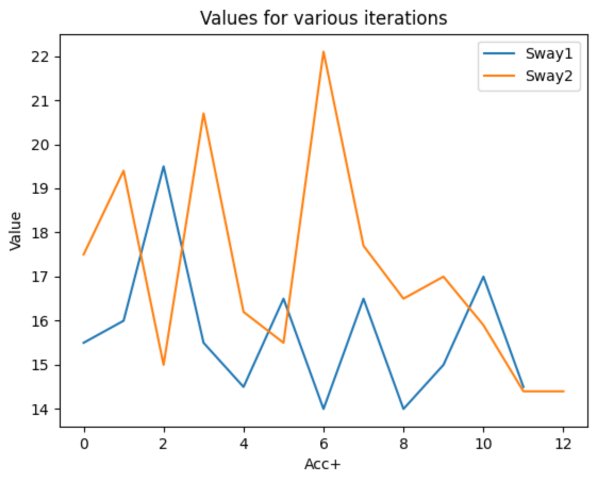

Fig. 10. Values for Acc: Auto 93

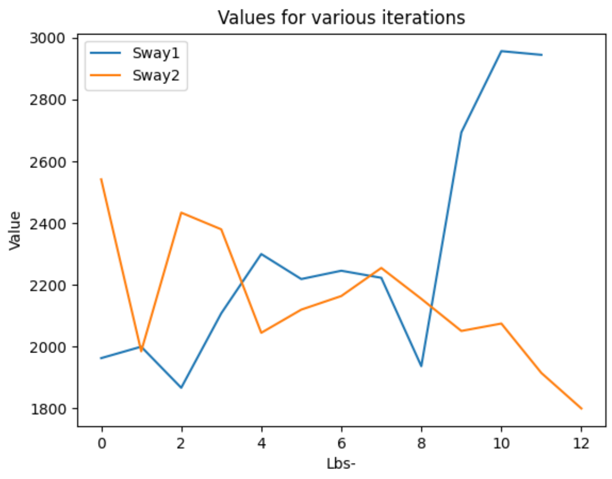

Fig. 11. Values for LBS: Auto 93

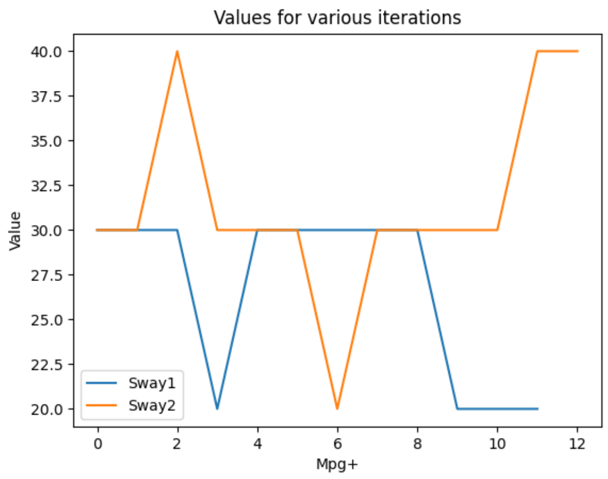

Fig. 12. Values for MPG: Auto 93

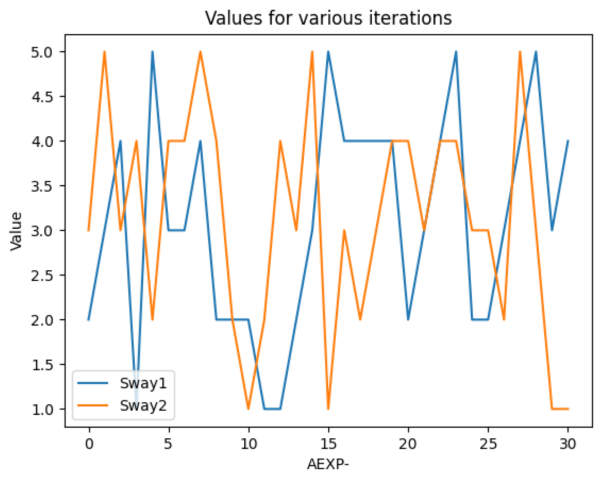

Fig. 13. Values for AEXP: COC1000

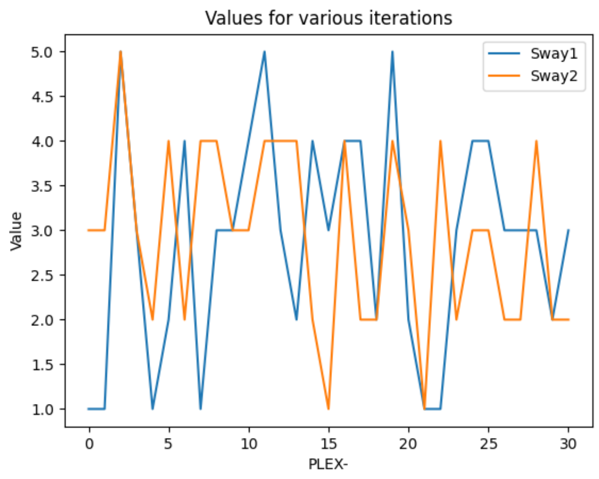

Fig. 16. Values for PLEX: COC1000

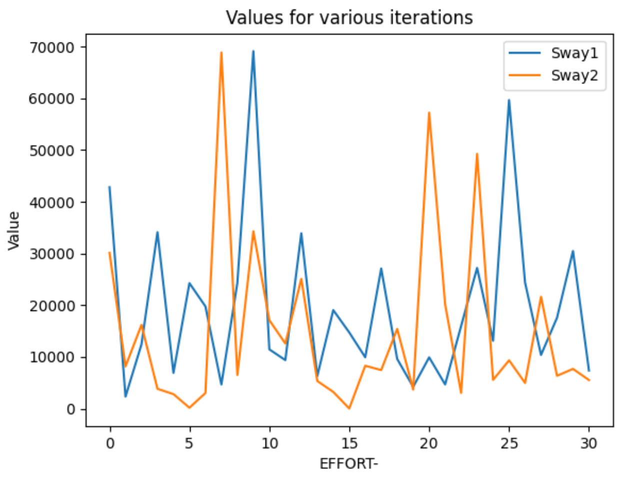

Fig. 14. Values for Effort: COC1000

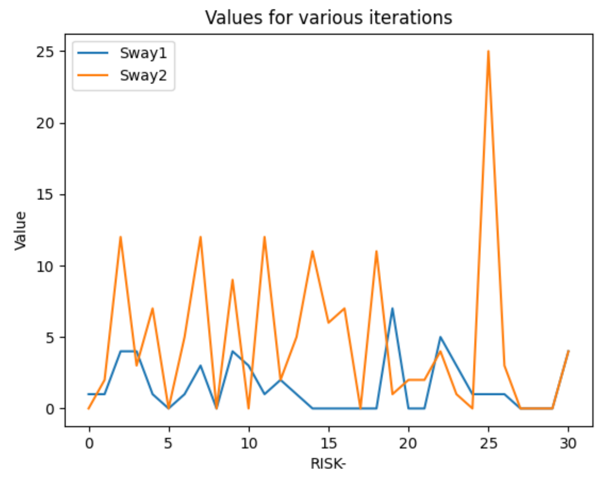

Fig. 17. Values for Risk: COC1000

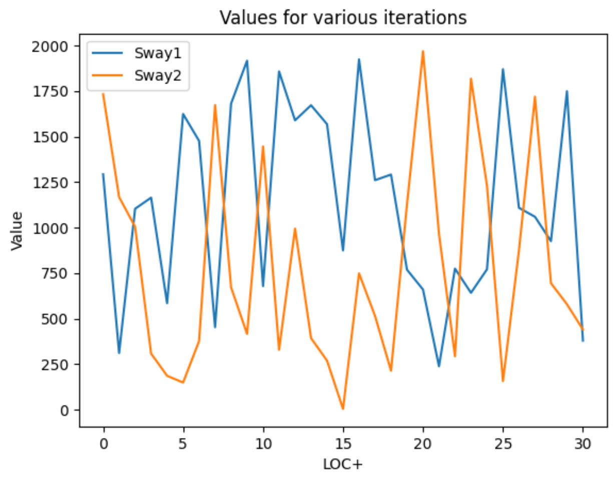

Fig. 15. Values for LOC: COC1000

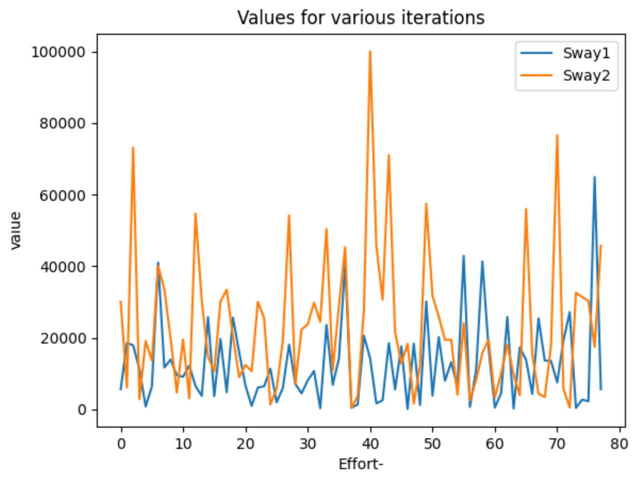

Fig. 18. Values for Effort: COC10000

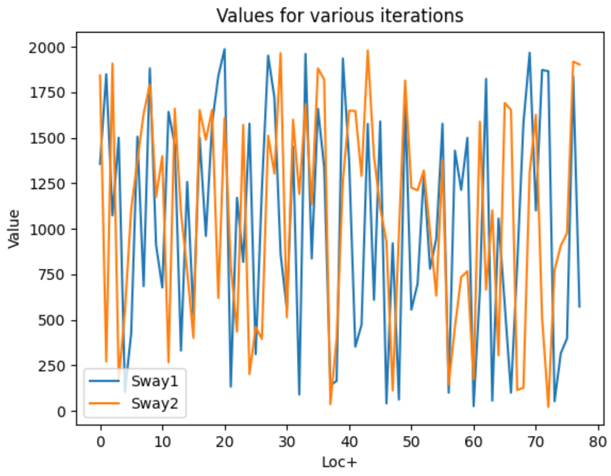

Fig. 19. Values for LOC: COC10000

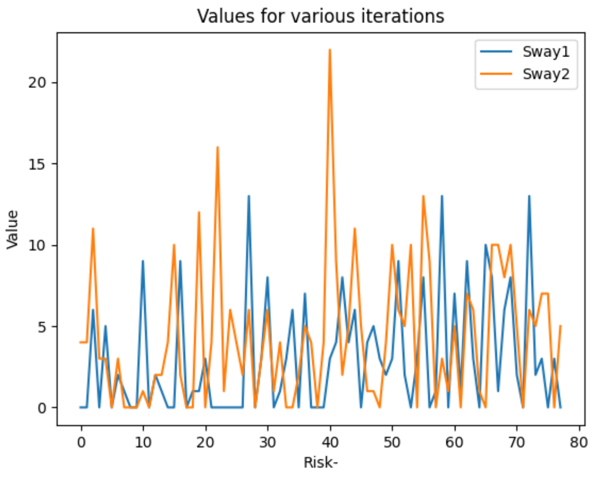

Fig. 20. Values for EffRiskort: COC10000