Multi-objective Semi-supervised Explanation System

Kaksha Mhatre*, Janvi Jitendra Phadtare†, Saail Ganesh‡ and Sourabh Pardeshi§
Department of Computer Science
North Carolina State University, Raleigh, NC 27606
Emails: * kpmhatre@ncsu.edu, † jphadta@ncsu.edu, ‡ sganesh@ncsu.edu, § sjpardes@ncsu.edu
Github: https://github.com/AutoSE/Project - Group 25

***Abstract*—Nowadays, search-based optimization techniques are being increasingly used by Software Engineering (SE) researchers to tackle SE problems that involve multiple conflicting objectives. Typically, these techniques involve utilizing CPU-intensive evolutionary algorithms to explore generations of mutations to a population of candidate solutions. However, an alternative approach has been proposed in this paper, which involves starting with a very large population and then sampling down to only the better solutions. We have tried to improvise on SWAY by focusing on the technique used to split the data into clusters. The paper compares SWAY to our new improvised SWAY. According to the experiments conducted in the paper, this new technique is found to be on par with SWAY. Given the simplicity and effectiveness of our algorithm, we propose it as a comparable model to SWAY for search-based software engineering models.**

I. Introduction

It is extremely difficult to develop a single algorithm that will be the best for all optimization problems. While algorithm A may perform best for one class of problems, it may perform poorly for another. Therefore, when implementing a new domain, some experimentation is always necessary to determine the appropriate algorithm for the specific model. This means that a lot of algorithms are local to a particular domain and fail to perform well on other domains. For this SWAY was created. SWAY serves as a baseline model that possesses all the attributes that are sought after in a baseline method, including straightforward implementation and speedy execution times.

For multi-objective optimization problems, evolutionary algorithms can be computationally intensive and time-consuming when compared to SWAY which is a simpler method that can be implemented quickly and has fast execution times. This suggests that SWAY may be a viable alternative to evolutionary algorithms in some cases, particularly when speed and simplicity are important considerations. Furthermore, SWAY requires very little additional effort when studying a new problem, making it a useful baseline method for optimization problems in software engineering and related fields. In summary, SWAY may be preferred over evolutionary algorithms in cases where simplicity and speed of implementation are more important than the absolute best possible performance, and where there is a need for a quick, low-cost baseline method to assess the feasibility of a new optimization problem.

Baseline models enable developers to efficiently eliminate any optimization option that performs below the minimum acceptable level. This allows researchers to quickly obtain preliminary results and obtain guidance for future experimentation. Our proposed model aims to optimize SWAY to implement an improvised baseline model. According to [1], we know that the fast termination of SWAY is traded off by the quality of the solution it provides, which is why SWAT is beaten by multi-objective evolutionary algorithms (MOEA) in some cases. We explore the possibility of improving the accuracy of the SPLIT function in SWAY, which would give us a better quality solution without compromising the speed of execution.

In our proposed model, we have tried to improve the SPLIT function of SWAY by comparing the results of various clustering techniques like Agglomerative Clustering, DBSCAN, BIRCH, and Spectral Clustering as opposed to the FastMap heuristic explained in [1]. With our improvised SPLIT function in SWAY, we have aimed to minimize the trade-off between the quality of the solution and the fast performance of the original SWAY. Here, we still follow the SWAY principles of clustering the candidates based on the decisions that than the objectives.

A. Structure of this paper

The structure of this paper is organized as follows:

In the following part of this section, the main contributors of this paper will be introduced. Section 2 introduces the related work of other researchers which provides a crucial introduction to the method that this paper tries to improvise.

The first part of Section 3 will provide information about the existing methods that are used for multi-objective optimization, and the second part of Section 3 will elaborate on the tried and tested methods to improve the SPLIT function of the original SWAY and the overall comparison of these experiments.

Section 4 will showcase the results of our experiments conducted in Section 3.

In Section 5, we will arrive at a conclusion and select the technique that provides the most improvisation on the original SWAY, based on the statistical results of our experiments.

We collaborated on the following research questions while working on this project:
1) Is it possible to create a novel algorithm using Deep Reinforcement Learning for Multi-Objective Optimization (DRL-MOA)? [2]
2) Is there a way to optimize the original SWAY to provide a better baseline algorithm?
3) What areas could be optimized in the original SWAY (Sampling method, Using other distance metrics, Data splitting for clustering, etc)?
4) Which would be the most efficient improvisation without trading off the execution time that the original SWAY provides?

Firstly, we tried to think of a novel solution to multi-objective optimization by using DRL-MOA which employs deep reinforcement learning (DRL) to solve the multi-objective optimization problem (MOP) in a simple and effective manner. A major caveat to the above approach was the time constraint for the project as we thought it was not viable to implement the code with significant improvements.

Hence, to make the most out of the time we have, we thought of improving the original SWAY by changing one or more components of it. After carefully brain-storming the areas of improvement in SWAY by comparing the trade-off between the execution time of the algorithm and the quality of the solution, we finalized implementing code to improvise the SPLIT method in the original SWAY which is responsible for splitting the data into clusters.

We performed an exhaustive literature survey to find the best clustering technique that would provide a significant improvement over the existing SPLIT method which used the FastMap heuristic.

Our paper encompasses different ways of clustering that could enhance the performance of the original SWAY without compromising its time performance which is the primary goal of the original SWAY, thereby providing a better baseline model.

The unique contributions of this paper are as follows:
1) We tried to improvise the SPLIT function of the original SWAY.
2) A single SPLIT function works for both continuous and discrete variables as opposed to the original SWAY that required two separate SPLIT functions.
3) Results are evaluated by the metrics like Kruskal Wallis Test, Scott Knott Test, Wilcoxon Rank Sum Test, and Mann Whitney Test.

II. Related Work

This paper represents a substantial expansion of the authors' previous research efforts in [1]. In [1], SWAY is proposed as the baseline optimizer for search-based SE problems. It discusses two approaches for the SPLIT function, one for continuous space (FastMap) and one for binary decision spaces.

A. FastMap Heuristic for continuous spaces

The SPLIT algorithm partitions candidates into subsets and selects representatives for each subset. To efficiently split candidates with continuous decisions, they utilize the FastMap heuristic. The FastMap heuristic is utilized to split candidates into two subsets by selecting a random candidate, locating the two extreme candidates based on distances, projecting all other candidates onto the line between these extremes, and then splitting the candidates into two subsets based on their projections. Their case studies employ the Euclidean distance as the DISTANCE metric.[1][3][4]

B. Handling Binary Decision spaces

In [1], a separate SPLIT algorithm had to be written for problems with binary decisions. This is because, research in [1] discovered that in a D-dimensional decision space where each decision can only take one of two binary values (i.e., 0 or 1), all candidates would be located at the vertices of the space. As a result, the continuous version of the SPLIT algorithm, which was previously described, failed to perform effectively as SWAY (the decision-making component of the algorithm) repeatedly suggested divisions of the empty spaces between the vertices, which were eventually of no use.

Another paper [2] that we referred to had a more advanced approach. This paper presents a framework for using Deep Reinforcement Learning to solve multi-objective optimization problems, which involves training a neural network agent to learn a policy for selecting actions that balance the trade-offs between different objectives. The authors of this paper demonstrate that their approach outperforms traditional methods in terms of finding Pareto-optimal solutions and achieving a better balance between different objectives.

To achieve this, they first adopt a decomposition strategy (as presented in [6]) to divide the MOP into multiple sub-problems, where each sub-problem is represented as a neural network. Then, a collaborative optimization technique is used, which combines the neighborhood-based parameter transfer strategy and the Actor-Critic training algorithm [7], to optimize the model parameters of all sub-problems.

However, this method had disadvantages of its own. Deep Reinforcement Learning algorithms are complex and require significant computational resources and time to train. These algorithms also tend to struggle in new or unseen environments. This made this approach less suitable for our objective where the problem changes over time or where the agent must adapt to new situations.

After exploring the prior work mentioned in the above section and taking into consideration the time constraints of the project, we decided to work on finding an alternative for the FastMap Heuristic and optimize the SPLIT function of the original SWAY. We were able to keep the essence of SWAY, i.e. we did not modify the sampling mechanism, we clustered candidates based on the decisions rather than the objectives, because if we cluster the candidates through their objectives,

we would need to evaluate all candidates [1] which would introduce slowness to the algorithm.

The part we did modify was trying out clustering techniques like BIRCH, Agglomerative Clustering, and Spectral Clustering. We chose to explore this because the FastMap heuristic although very time efficient, fails to provide scope for scalability, and is sensitive to the choice of distance metrics used, that is, it may not work well with a wide range of distance metrics. Moreover, FastMap produces a lower-dimensional representation of the data, which can be difficult to interpret without prior knowledge of the data. In the sections below, we have explained how we found that some of these clustering techniques successfully outperformed the original SPLIT function in SWAY.

III. Methods

In this section, we explain the methods that we implemented to improve the results of SWAY. The SWAY method has two main components: better and half. The better function uses Zitzler over boolean domination due to its obvious advantage as it allows for more flexibility in comparing and ranking solutions based on their performance on multiple objectives. It also allows for the possibility of non-dominated solutions that do not satisfy all constraints, which can be useful in certain optimization scenarios. Hence, we found a little scope for improvement there. A better approach was to try and improve the half function by generating better clusters. We implemented the following mentioned algorithms to achieve the same.

A. Algorithms

1) BIRCH: Among the other Balanced Iterative Reducing and Clustering using Hierarchies(BIRCH) [5] was used to split the data. It works by constructing a hierarchical clustering structure, where each node in the hierarchy represents a cluster. The algorithm uses a technique called Clustering Feature Tree (CFT), which is a tree-like data structure that summarizes the distribution of data points in a cluster. The CFT is used to incrementally merge clusters in a bottom-up fashion, starting from the leaves of the tree and moving toward the root. The resulting hierarchy is output as a tree, where each leaf node represents a cluster and the root represents the entire dataset.

The idea behind using BIRCH was simple, it has low memory requirements and arrives at the results fast. It can also handle large datasets efficiently. However, since BIRCH is designed for datasets with a relatively uniform distribution and does not perform well on datasets with complex structures or varying densities.

2) Agglomerative Clustering: In multi-objective optimization, the goal is to simultaneously optimize multiple objectives that may be conflicting or incompatible. Agglomerative clustering starts with each data point as a separate cluster and then iteratively merges pairs of clusters based on their similarity until all data points belong to a single cluster. Each solution in the optimization problem is a data point in a high-dimensional space, with each objective representing a dimension.

The clustering process results in a hierarchical structure of clusters, with each cluster representing a set of solutions that are similar to each other in terms of their objective values. It helps to reduce the search space and focus on the most promising solutions for multi-objective optimization problems. The advantage of Ward's linkage is that it tends to produce compact, spherical clusters of roughly equal size. By minimizing this objective, Ward's linkage can produce clusters that are more tightly packed and more clearly separated from one another than other linkage methods.

3) Spectral Clustering: Since the Spectral clustering method is less sensitive to outliers and works well in high dimensional space and on clusters of varied shapes on which the traditional algorithms like k-means fail, we attempted at improving the half function using this algorithm. It uses the spectrum (eigenvalues) of a similarity matrix to cluster data points into groups.

To apply spectral clustering, the initial step is to create a similarity matrix that reflects the pairwise similarities between data points. Various similarity measures can be employed, including the Gaussian kernel, cosine similarity, or Euclidean distance. Once the similarity matrix is formed, eigenvalues and eigenvectors are computed utilizing linear algebra techniques. The eigenvectors associated with the top eigenvalues are employed to project the data points into a low-dimensional space. This low-dimensional representation can then be clustered by a conventional clustering algorithm.

Though spectral clustering seemed an ideal choice at first instance, we observed it to be computationally expensive for large datasets especially while calculating the eigenvalues and eigenvectors and selecting optimal parameters was quite challenging.

B. Data

Dataset	Datapoints	Columns
auto2.csv	93	23
auto93.csv	499	19
china.csv	499	19
coc1000.csv	1000	25
coc.10000.csv	10000	25
healthCloseIsses12mths0001-hard.csv	10000	8
healthCloseIsses12mths0011-easy.csv	10000	8
nasa93dem.csv	93	29
pom.csv	10000	13
SSM.csv	239360	19
SSN.csv	53662	19

Fig1: Data

1) auto2.csv: This car dataset has target attributes like MPG, weight, and class based on predictor attributes like

type, RPM, etc. The objective is to maximize CityMPG, and highwayMPG and minimize weight and class. The most correlated attributes are engine size, horsepower, fuel tank capacity, and width.

2) auto93.csv: Here acc and mpg need to be maximized and lbs need to be minimized. The most correlated attributes are Volume and Clnds.

3) china.csv: The target variable N-effort needs to be minimum. The effort, AFX, and added are the most related attributes.

4) coc1000.csv: LOC needs to be high and AEXP, PLEX, RISK, and EFFORT need to be minimum. The most correlated attributes are ACAP, PCAP, SCED.

5) coc10000.csv: Loc has to be maximum and Effort and Risk need to be minimized. Attributes with maximum correlation: Acap, Pcap and Sced

6) healthCloseIsses12mths0001-hard.csv: ACC and PRED40 are to be maximized and MRE minimized.

7) healthCloseIsses12mths0011-easy.csv: ACC and PRED40 are to be maximized and MRE minimized. Attributes with maximum correlation: N_estimators

8) nasa93dem.csv: Maximize Kloc and minimize Efforts, Defects, and Months. Attributes that are most related are idX, centerX, and YearX.

9) pom.csv: Maximize Completion and minimize Cost and Idle. Attributes with maximum correlation: Criticality, Dynamism, and Size

10) SSM.csv: Minimize numberiterations and timetosolution. Attributes with maximum correlation: jACOBI

11) SSN.csv: Minimize PSNR and Energy. The most correlated attributes are no_cabac and Seek

IV. Results

After implementing alternatives to the SPLIT function in SWAY, we compared the results using various tests. Some datasets performed better with the original SWAY function, while some showed significant improvement with sway1(Agglomerative Clustering) or sway2(BIRCH). The changes made to SWAY demonstrate slight improvement in some datasets than the original algorithm.

The Kruskal-Wallis test is a statistical method that is used to compare three or more independent groups when the data do not meet the assumptions of normality and equal variances required by traditional ANOVA methods. It calculates a test statistic based on the ranks of all the observations across all groups. The test compares the medians of the groups and determines if there is a significant difference between them. If the test statistic is significant, it indicates that there is evidence to reject the null hypothesis, and at least one group is significantly different from the others.

The datasets were iterated 20 times and each iteration provided us with the mean of the cluster after running the original SWAY, sway1, and sway2.

MWU and KW significance level: 0.05

	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway	2205.3	16.66	30.84	6
sway1	1798.26	17.15	34.74	4
sway2	1823.33	17.22	34.55	4
top	2970.42	15.57	23.84	398
samp tax1				
KW Sway p-vals	1.4489223590713 95e-08	0.0031322661101375717	6.589891632183954e-05	
Mann-Whitney U Sways	['sway1']	['sway2']	['sway1']	
Kruskal-Wallis Sways	['sway1']	['sway2']	['sway1']	
samp tax2				
KW Sway p-vals	1.4489223590713 95e-08	0.0031322661101375717	6.589891632183954e-05	
Mann-Whitney U Sways	['sway1']	['sway2']	['sway1']	

Fig2: Output for auto93.csv

We ran all three SWAY algorithms on the auto93.csv dataset. We found that overall sway1 had better results than sway2 and SWAY. We found that agglomerative clustering had improved the results for auto93.csv

MWU and KW significance level: 0.05

	N_effort-	n_evals avg
all	4277.64	0
sway	2371.81	6
sway1	191.96	7
sway2	206.3	7
top	4277.64	499
samp tax1		
KW Sway p-vals	3.6846795313032695e-08	
Mann-Whitney U Sways	['sway1']	
Kruskal-Wallis Sways	['sway1']	
samp tax2		
KW Sway p-vals	3.6846795313032695e-08	
Mann-Whitney U Sways	['sway1']	
Kruskal-Wallis Sways	['sway1']	
samp tax3		
KW Sway p-vals	3.6846795313032695e-08	
Mann-Whitney U Sways	['sway1']	
Kruskal-Wallis Sways	['sway1']	
samp tax4		
KW Sway p-vals	3.6846795313032695e-08	
Mann-Whitney U Sways	['sway1']	
Kruskal-Wallis Sways	['sway1']	

	N_effort-
all to all	=
all to sway	≠
sway to sway1	≠
sway to sway2	≠
sway1 to sway2	≠
sway1 to top	≠

Fig3: Output for china.csv

For china.csv, sway1 performed better in Kruskal Wallis Test. In this dataset, as there is just one minimization, we can see that the lowest value for N_effort- would be the best value. Agglomerative has performed the best.

MWU and KW significance level: 0.05

	Cost-	Completion+	Idle-	n_evals avg
all	369.99	0.87	0.24	0
sway	243.5	0.86	0.25	8
sway1	172.03	0.86	0.25	6
sway2	181.69	0.86	0.23	7
top	369.99	0.87	0.24	10000
samp tax1				
KW Sway p-vals	1.0682321512778716e-46	0.0005355334608562384	0.0036988056699299794	
Mann-Whitney U Sways	['sway1']	['sway']	['sway2']	
Kruskal-Wallis Sways	['sway1']	['sway']	['sway2']	
samp tax2				
KW Sway p-vals	1.0682321512778716e-46	0.0005355334608562384	0.0036988056699299794	
Mann-Whitney U Sways	['sway1']	['sway']	['sway2']	
Kruskal-Wallis Sways	['sway1']	['sway']	['sway2']	
samp tax3				
KW Sway p-vals	1.0682321512778716e-46	0.0005355334608562384	0.0036988056699299794	

Fig4: Output for pom.csv

V. Discussion

The SWAY algorithm might not provide the best solution but it provides a simple, fast, and efficient solution.

A. Threats to Validity

One of the threats to the validity of the above-proposed methods of implementation is the assumption that the Zitzler predicate is the best possible option to choose the best cluster. The choice of the linkage method also has a significant impact on the quality of the clustering results. The usage of the ward's linkage is based on the assumption that the data is Euclidean. In case we get a dataset that has a binary or categorical target, the ward's linkage may not work well. In addition, agglomerative models are prone to the risk of overfitting which can cause lead to a poor baseline because of poor generalization.

B. Future Work

This study primarily focuses on enhancing the "half" algorithm of SWAY, although there can be future research that demonstrates that the "better" algorithm can also be enhanced in order to enhance SWAY. Although the scope of our research was restricted to using Zitzler's predicate, it is still possible that if anything better than Zitzler's predicate were to be discovered, it might be utilized to enhance the "better" method, so enhancing the SWAY.

VI. Conclusion

In this experiment, we found different approaches like Hierarchical clustering to improvise SWAY. We compared various clustering algorithms and found algorithms that provide a better baseline model than SWAY.

While the algorithms performed better than the original SWAY for most of the datasets, the SPLIT function in the original SWAY performed better for a few datasets that were used for testing. We compared Agglomerative and BIRCH with 10 different datasets of different shapes and sizes and found that Agglomerative and BIRCH are resistant to an increase in sampled data. With the increase in sampled data, both algorithms have performed better than the original SPLIT function.

With metrics, we can conclude that Agglomerative and BIRCH have a slight improvement over the original SWAY algorithm.

VII. Bonus Section

A. February Study

In the beginning, we tried other approaches to tackle the same problem as SWAY in an effort to improve it. The primary goal of the challenge was to carry out multi-objective optimization and attempt to enhance SWAY results. Utilizing reinforcement learning to optimize various objectives was one strategy we came up with. The idea of awarding points for achieving one goal optimization and penalizing for the contrary can result in a model that can achieve multiple goal optimization. After some research on this idea, we came across the paper [2] "Deep Reinforcement Learning for Multi-objective Optimization". This paper discussed that solving each scalar optimization usually leads to Pareto optimal Solution and the desired Pareto Front(PF) can be obtained when all the scalar optimizations are solved. This Pareto Front can be formed by using the solutions obtained by N sub-problems using the Neighbourhood-based parameter-transfer strategy.

However, when we went back to the high-level problem statement, we realized that using Deep Reinforcement Learning would be a very complex (and expensive) approach to solving the problem. Instead, we can solve the problem statement for a low cost by attempting to improve the structure of the provided algorithm SWAY or by attempting to improve the integral parts of SWAY, such as better and half, in order to improve the outcomes produced by SWAY.

Our February study came to the conclusion that using the available resources would allow us to answer the problem statement more cheaply than developing a brand-new methodology.

Acknowledgment

We would like to express our sincere gratitude to Dr. Timothy Menzies, for his invaluable guidance, support, and encouragement throughout the course of this research. His extensive knowledge, insightful comments, and constructive criticism have been instrumental in shaping our ideas and refining the contents of this paper.

We would also like to thank the teaching assistants, Andre Lustosa, Rahul Yedida, and Xueqi(Sherry) Yang, for their tireless efforts in providing feedback and assistance throughout the research process. Their prompt responses, insightful suggestions, and attention to detail have been crucial in helping us improve the quality of our work.

Finally, we are grateful to all our fellow classmates and researchers mentioned in our references who generously shared their time, expertise, and resources with us during the course of this study.

Thank you all for your contributions, and for making this project possible.

REFERENCES

[1] Chen, J., Nair, V., Krishna, R., Menzies, T. (2018). "Sampling" as a Baseline Optimizer for Search-based Software Engineering. IEEE Transactions on Software Engineering, 1–1. doi:10.1109/tse.2018.2790925

[2] K. Li, T. Zhang and R. Wang, "Deep Reinforcement Learning for Multiobjective Optimization," in IEEE Transactions on Cybernetics, vol. 51, no. 6, pp. 3103-3114, June 2021, doi: 10.1109/TCYB.2020.2977661.

[3] Christos Faloutsos and King-Ip Lin. Fastmap: a fast algorithm for indexing, datamining and visualization of traditional and multimedia datasets. In ACM SIGMOD international conference on Management of data, 1995

[4] John C. Platt. Fastmap, metricmap, and landmark MDS are all nystrom algorithms. In Proceedings of 10th International Workshop on Artificial Intelligence and Statistics, pages 261–268, 2005.

[5] Kovacs L, Bednarik L, et al. Parameter optimization for BIRCH preclustering algorithm. IEEE International Symposium on Computational Intelligence and Informatics; 2011

[6] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," IEEE Transactions on evolutionary computation, vol. 11, no. 6, pp. 712–731, 2007.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in International conference on machine learning, 2016, pp. 1928–1937.