Multiobjective Optimization Using Different Semi-supervised Clustering Approaches

Arun Srinivasan Parthasarathy
College of Engineering
Computer Science
North Carolina State University
Raleigh, NC 27606
Email:apartha4@ncsu.edu
Karan Pradeep Gala
College of Engineering
Computer Science
Raleigh, USA
Email: kgala2@ncsu.edu
Vishnu Vinod Erapalli
College of Engineering
Computer Science
Raleigh, 27606
Email: verapal@ncsu.edu

***Abstract*—Often problems have multiple objectives and each one needs to be taken care off simultaneously. We would have to maximise some while keeping the others constant or minimize one while maximising the others. There are many such combinations that exist and the task of balancing these multiple objectives in order to move closer to our goal is known as multiobjective optimization. In modern day and age, multiobjective optimization plays a crucial role and to tackle this, clustering is a feasible solution. With inspiration drawn from SWAY, this paper attempts to improve the existing method to try and compete with the results. SWAY generates BEST and REST clusters and the decider for this is Zitzler's domination predicate. This paper uses techniques such as dimensionality reduction, semi-supervised clustering, and rule generation in order to solve the optimization problem. Explanation algorithms use rule generation in order to textually describe the clustering algorithm. This paper also aims to change the way rules are generated in order to assess the improvement in results. To check the significance in results obtained, this study uses a non-parametric effect size test and a non-parametric significance test .**

I. Introduction

Multi-objective optimization is a mathematical optimization technique that involves optimizing multiple objectives simultaneously. The aim is to identify a set of solutions that are optimal with respect to multiple conflicting objectives, rather than just one objective as in single-objective optimization. These optimal solutions are those that are not dominated by any other solution in the set, known as Pareto-optimal solutions. The Pareto front, which is the set of all Pareto-optimal solutions, is identified in multi-objective optimization. This technique has diverse applications in fields such as engineering, finance, logistics, and environmental management. To solve this multi-objective optimization problem, we learn about the SWAY algorithm. When there are multiple Ys, we need to keep in mind that it is expensive to access them. Thus we constrain ourselves to access as little as possible. During the course, we learn about an algorithm that tries to tackle this problem by performing clustering. The output of this algorithm is a 'better' cluster which shows that a point picked from here might yield you a better solution. In this approach, we first look at all the points in space. Since there are n attributes, the dimension will also be nd. Instead of accessing all the scattered points in space, we restrict ourselves to dealing with only a certain number of points within a given boundary. This is decided by the.far and not only does this decide the boundary but it also protects us from outliers. Once this is done, we pick a random point in the cluster and find a point furthest from it. From this new point, we find another furthest point and draw a line. All points in space are now projected onto this line. We observe that this technique eventually helped us reduce dimensionality too. We now select the better half by using Zitzlers domination. We discard the other half and recurse on the better half. As opposed to many divide-and-conquer approaches, this goes down only the optimal path and does not recurse over the other half.

One major drawback of the clustering method is that it heavily relies on the initial random point selection, which can lead to sub-optimal or biased solutions. Additionally, the method can be sensitive to the order in which points are selected, as the furthest point may not always be the best representative of the cluster. Furthermore, the projection onto the line between points B and C can also result in a loss of information, as it collapses the multi-dimensional space onto a single dimension. This can lead to the grouping of dissimilar points and the separation of similar points, which may not be desirable. One of the main drawbacks of using Zitzler's domination metric is that it assumes a linear relationship between objectives, which may not be the case in all optimization problems. This can result in sub-optimal solutions being chosen, as the metric may not accurately reflect the true preferences of the decision-maker. Zitzler's domination metric can be computationally expensive, particularly for large-scale multi-objective optimization problems with many objectives and/or solutions. This can limit its practical use in certain applications.

We feel that it is wise to change the clustering algorithm in order to try and compete with the state of the art algorithm,i.e, SWAY. We rely on agglomerative clustering [1] and it comes with certain added benefits. First, agglomerative clustering does not require a random point selection, which can lead to biased or sub-optimal solutions. Instead, it starts with all points as individual clusters and merges them based on a similarity measure. This ensures that all points are considered and can potentially be part of the final clusters. Agglomerative clustering does not rely on the projection of points onto a single line or dimension, which can lead to a loss of information and potentially group dissimilar points or separate similar points. Also, Agglomerative clustering can incorporate different similarity measures, such as Euclidean distance, cosine similarity, or correlation, which can better capture the underlying structure of the data and the preferences of the decision-maker. Finally, agglomerative clustering can be used with different stopping criteria, such as a fixed number of clusters, a threshold on the similarity measure, or a visual inspection of the dendogram, which can provide more flexibility and control over the clustering process.

We have another approach in mind to solve the multi-objective optimization problem. SWAY relies on the worker function which in turn relies on the half function which performs clustering. Our proposal for half function revolves around agglomerative clustering.

A. Structure of this paper

There are several research questions that arise with respect to our topic of discussion. How can we use clustering to generate a diverse set of solutions in a multi-objective optimization problem? One technique would be to cluster the existing set of solutions in the problem space into distinct groups, and then select one representative solution from each cluster to form a diverse set of solutions.The next question that arises is: What is the impact of different clustering algorithms on the quality of solutions obtained in a multi-objective optimization problem? We aim to answer this by trying out different variants of SWAY ourselves. The next question would be: How can we use clustering to reduce the number of solutions that need to be evaluated in a multi-objective optimization problem? We believe that a strong stopping criteria or a good recursive implementation with the right hyperparameter values would be a good way to reduce the

number of evaluations. Last but not the least, how can we group similar solutions in a multi-objective optimization problem to better understand the trade-offs between objectives? This could be solved using a similarity measure which compares two solutions and decides how close they are in multidimensional space and one of the best known techniques would be to encode them into vectors and pass them through a cosine similarity function.

This paper talks about how we use our pre-existing knowledge on SWAY, half, better, and Zitzler's domination and come up with an alternative approach to clustering in order to tackle the problem of multi-objective optimization. This paper also talks about the results obtained when we tweak the core algorithm of SWAY. With an in-depth analysis of SWAY and xpln, we present the performance of our algorithm when run on 11 different datasets and compare this against the performance on SWAY. We also come up with a modification to xpln and we call it xpln2 which aims to improve the explanation algorithm which in turn generates some rules. The paper speaks for itself since it not only explores the changes in the core functionality but it also tries to challenge and surpass the performance of SWAY.

We know that the datasets provided, all have certain attributes which are to be maximized or minimized and we call these the Y's. We also have independent attributes which are called the X's. During our clustering, we tend to compute the centroids and in the process of doing so, we should keep in mind that we should not involve Y in the computation of the centroids. That is, when we compute the arithmetic mean to obtain the centroids, we should not involve the dependent variables. However, in our computation, to make it less complex and in order to decrease the number of lookups (since Y lookups are expensive), we resorted to involving Ys also in the computation of centroids. This is one of the caveats that stand out in our implementation.

II. Related Work

Random projection can be a useful technique for dimensionality reduction, its effectiveness for clustering depends on the specific dataset and the quality of the random projection used which can make it difficult to find optimal solutions. Random projection can be used to map the objective functions onto a lower-dimensional subspace while preserving the important features of the original functions. It is important to carefully consider the limitations and assumptions of the approach and validate its effectiveness on a variety of datasets. Random projection assumes that the data can be well-separated in the projected space, which may not always be the case in real-world datasets. In some cases, the data may be intrinsically high-dimensional and cannot be easily separated in a lower-dimensional space, which can make clustering more challenging. As mentioned earlier random projection can sometimes result in a loss of information, as it projects the data onto a lower-dimensional space and can discard some of the original features.

Semi-supervised learning can be a powerful tool for multi-objective optimization. It leverages both labeled and unlabeled solutions and incorporates domain knowledge or preferences and reduces the number of evaluations needed to find optimal solutions. In particular, the Semi-supervised clustering approach tends to group similar solutions together based on the features or objective function values. The best part is that it uses the labeled solutions to refine the clustering or classification. It gets the best of both worlds of supervised and unsupervised learning. This helps us identify clusters of solutions that are likely to be optimal and it focuses its search on those regions of space. Furthermore, we talk a lot about evaluations and SWAY runs many evaluations too. On that note, during unsupervised learning, the classifier or surrogate model would be trained using the labeled solutions and then it can be used to predict the quality of new solutions without actually evaluating their objective function values. This can reduce the number of evaluations needed to find optimal solutions, as the classifier or surrogate model can be used to guide the search toward promising regions of the search space.

From the paper on 'Why heuristics work,' we learned heuristics are problem-solving methods that do not guarantee an optimal solution but aim to find a good enough solution that satisfies certain criteria or constraints. Heuristics are often used in situations where it is not feasible or practical to exhaustively search the entire solution space, or where the problem is too complex to be solved by traditional optimization methods. Also, we found out that when using heuristics, the goal is to find a solution that meets a certain level of acceptability, rather than trying to achieve the best possible outcome. This is similar to setting an aspiration level, which is the minimum level of acceptability for a solution. On similar grounds, with our sway2 we aim to satisfice ,i.e, find a good enough solution as opposed to finding the best solution. When you present a cluster as a solution for an optimization problem, it basically portrays that you are presenting a bunch of candidate solutions and one of them can lead to an amazing solution.

Moreover, we observed that explanation plays a very important role in multi-objective optimization, and with xpln as an inspiration we dived deep into understanding what makes it necessary to use an explanation algorithm. We learned that in a multi-objective scenario, a model is trained to optimize several objectives simultaneously and the model's decision-making process may be difficult to interpret, making it hard to understand why it made a particular prediction. To better understand it, we would want to provide explanations for the model's predictions or decisions. We should remember that the model is trained on multiple objectives which might have different importance from each other and in real-world scenarios, we have access to only a limited amount of labeled data and we would have to rely on the unlabeled data to provide the explanations. In semi-supervised learning, we have both labeled and unlabeled data and the goal of the explanation for Multi-objective optimization with the semi-supervised model is to provide explanations for this model which is trained using semi-supervised learning. This requires balancing the trade-off between different objectives and the amount of labeled and unlabeled data that is available to us. All in all, it seems that this is a challenging area in research that requires us to address multiple objectives leveraging both labeled and unlabeled data in order to provide accurate and interpretable explanations for complex models.

From our understanding of the papers and topics and keeping in mind the professor's method of tackling multi-objective optimization, we proceeded to use the same approach to solving this problem. We too decided to go ahead with clustering as we felt that clustering is a good approach to multi-objective optimization because it can help simplify complex multi-objective problems by grouping similar solutions together. We aim to find a set of solutions that optimize multiple objectives simultaneously but as the number of objectives increases, the search space becomes more complex and the problem of finding an optimal solution becomes more challenging. If we use Clustering. it can help to address this issue by grouping similar solutions together into clusters. By doing so, we can reduce the search space and focus on optimizing within each cluster. In addition to this, it can help identify the trade-offs between different objectives. By clustering solutions, we can identify which objectives are most important within each cluster and focus on optimizing those objectives. Also, we could grasp that clustering can identify Pareto optimal solutions within a cluster and we can then use these solutions as a starting point for further optimization.

Looking at the data provided to us, we observed that it involves

a good amount of Xs and a good amount of Ys. We believe that semi-supervised clustering is itself a state-of-the-art technique for solving multi-objective optimization problems and we look to tweak the internal working and core implementation in an attempt to improve the results and functionality. These techniques would work in most cases but we do have to point out that they can fail if there is poor data quality. Semi-supervised clustering relies on having a small set of labeled data and a larger set of unlabeled data. If the labeled data is of poor quality, with inaccurate or inconsistent labels, the clustering results may be unreliable and fail to capture the true underlying patterns. It also fails if there is high dimensionality in data. Moreover, Lack of separability between clusters, Conflicting objectives and Biased or incomplete representation of data are some cases where semi-supervised clustering might fail. However, we feel that these state-of-the-art techniques still do more good than bad since it is evident that there is an attempt to tackle and overcome these demerits. For example, we said that it will fail when there is high dimensional data but if we closely observe SWAY, it ensures to reduce the dimensionality first, and then it continues with doing the half. Not only does the method reduce dimensionality but it also does so in log-linear time as opposed to excellent methods like PCA. There is a lot here to draw inspiration from ! Therefore, from the advantages we learned from our research study, we decided to use semi-supervised clustering. Reflecting on all the information gathered, we feel that, what work has been done in this field has a lot of positivity to offer and we are looking to leverage the good parts that have come out of these.

III. Methods

A. Algo

The approach towards semi supervised clustering is dealt by SWAY. SWAY is a recursive clustering method takes in the data set and picks a random point. From this random point, it finds the furthest point. It marks the boundary that is given by the.far and within the boundary it picks another random point, say A. From A, it finds the furthest point say B. With A and B as a line, it projects all the points onto this line, thus reducing dimensionality. It then halves the points and chooses the better half using Zitzler's coefficient. Once the better half is chosen it recurses on the better half and it does not care about the discarded half. It continues until the last cluster that's resulted is of the order of 'the.min'. All in all, in less evaluations as compared to the baseline model, SWAY was able to generate a set of optimal solutions whose average would be reflected as a row.

The first part of our approach would be to run the existing SWAY model on all the 11 datasets provided and analyze the results on each of the dataset. It would be evident that SWAY performs best on certain kinds of datasets while it does pretty average on the other kinds of datasets. Once that is up and running, we would proceed to explore our own methods of clustering. The datasets would be pre-processed before subjecting it to any algorithmic analysis.

As we were baffled by the performance and complexity of the model of SWAY, we dived deep into its working. SWAY depends on a worker function that handles the recursion and as a part of this the 'half' function plays a key role in clustering. Here is where we felt that we could try other clustering algorithms and check if the resulting clustering could yield any better results.

As for the clustering, we used agglomerative clustering. It is a type of hierarchical clustering that starts by considering each data point as a separate cluster and then it iteratively merges the closest pair of clusters based on some similarity metric, until all the data points belong to a single cluster. A hyper-parameter to this function would be the linkage and we chose average linkage over single and complete linkage. Average linkage considers the average pair-wise distances of each observation of the two clusters. Average linkage clustering is more robust to noise or outliers in the data compared to single and complete linkage clustering. Single linkage clustering can produce long, thin clusters that are highly sensitive to noise, while average linkage clustering calculates the average distance between all pairs of points, which can help reduce the effect of noise. Since our data is spread all over space, we cannot really comment whether there are outliers in our data, but it is always beneficial to safeguard ourselves. Our clustering too results in a left and a right half and the better half is decided by the 'better' function which was the case with SWAY. Not only do we return the two clusters as a result of our clustering, but we also return two centroids of the clusters, i.e, A and B. The steps to the algorithm are as follows:

1. Call SWAY2
2. Call the worker of SWAY2
3. Worker uses Half2, i.e, our clustering algorithms

3a) Start with each data point as a separate cluster.
3b) Compute the pairwise distances between all the clusters.
3c) Merge the two closest clusters into a new cluster.
3d) Recompute the pairwise distances between the new cluster and all the remaining clusters.
3e) Repeat steps c and d until all the data points belong to a single cluster. This algorithm is taken care by the sklearn library.

The algorithm returns two clusters and we additionally return two representative points and call them the centroids.

4. These points are passed to 'better' and via the zitzler domination predicate, the better centroid is picked.
5. The cluster with the better centroid is picked and is recursed on by the worker function.
6. End of the SWAY2 function returns a better and rest cluster. 7. Repeat the same for 20 runs with different random seeds.
8. Calculate mid of the outputs generated in all the runs and compare it with the baseline model.

In addition, we decided to work on the rule generation. As per the traditional rule generator, we have ranges that result from the SWAY's result. Based on these ranges, we have rules that are generated. Now each rule is assigned a value by using the 'value' function and we rank the rules. In the traditional approach, rule 1 is used and the combination of best and rest yielded is checked out. Next, rule 1 and 2 are put together and checked if this rule does any better by picking more bests and less or rests. We then append rule 3 to this and check the results yielded by this and so on. We found this rather strange since rule 2 never got a chance of its own ! It was always grouped with 1 and checked for, for its results. Similarly, rule 3 did not find a chance to act on its own but was coupled with rules 1 and 2 and then checked for. We thus resorted to tweaking xpln in such a way as to equip it to deal with a different set of rules. According to the professor's method if rules are ranked by their value in the given order, say: (rule1),(rule2),(rule3)...(ruleN), it is advised to group it in this manner: (rule1),(rule1 and rule2), (rule1 and rule2 and rule3),....,(rule1 and rule2 and rule3 and.... ruleN). We, however, decided to combine the rules and test for its performance in this fashion: For three rules rule1,rule2,rule3 arranged in order of decreasing ranks, we generate: (rule1),(rule1 and rule2), (rule1 and rule2 and rule3),(rule2), (rule2 and rule3), (rule3). This not only gives us a chance to test subsequent rules solely, but it also gives us a chance to combine the rules that are ranked 2nd and 3rd, which earlier was not the case. The first-ranked rule was always involved. We would call this xpln2.

We would then check for the performance of sway2 and xpln2 and check with the performance of SWAY and xpln and would then proceed to examine the improvements (if any). We would also aim to come up with a better clustering technique in order to leverage the use of 20 runs. If it is the case that the same cluster was generated in all 20 runs, the sway2 algorithm would not be contributing much to our feasibility analysis. We would thus have to work with random seeds for each run and tweak our sway2 algorithms. Moreover, another algorithm would be explored where it uses agglomerative clustering as well as dimensionality reduction to see if the reduction in the sparsity of data would do any good in optimizing our goals.

B. Data

Here is some of the datasets in tabular form.

	mean	std	min	max
Clndrs	5.471939	1.705783	3	8
Volume	194.4107	104.6452	68	455
HpX	104.4694	38.49116	46	230
Lbs-	2977.584	849.4026	1613	5140
Acc+	15.18622	2.7437	8	24
Model	75.97959	3.683737	70	82
origin	1.576531	0.805518	1	3
Mpg+	23.80102	8.346168	10	50

TABLE I: auto93

	mean	std	min	max
CityMPG+	23.08537	5.59365	16	46
HighwayMPG+	29.97561	5.011039	22	50
Air_Bags_standard	0.853659	0.72217	0	2
Drive_train_type	0.890244	0.471564	0	2
NUMber_of_cylinders	4.853659	1.296928	3	8
Engine_size	2.170732	1.003757	1	5
Horsepower	139.9512	51.05502	55	300
RPM	5328.049	583.7352	3800	6500
Engine_revolutions	2367.866	497.2828	1320	3755
manual_transmission	0.682927	0.4682	0	1
Fuel_tank_capacity	15.7439	3.102412	9	23
Passenger_capacity	4.939024	0.708808	4	6
Length	183.1585	15.27402	141	219
Wheelbase	103.2073	6.466892	90	117
Width	68.90244	3.693871	60	78
U-turn_space	38.62195	3.164776	32	45
Rear_seat_room	27.53659	2.841253	19	36
Luggage_capacity	13.89024	2.997967	6	22
Weight-	2988.171	565.9361	1695	4105
domestic	0.512195	0.502927	0	1
Class-	18.71951	9.897343	7	61

TABLE II: auto2

	mean	std	min	max
LOC+	1013.05	571.3543	4	1999
ACAP	3.07	1.251666	1	5
AEXP-	2.967	1.196387	1	5
ARCH	3.561	1.482726	1	6
CPLX	3.547	1.46695	1	6
DATA	3.456	0.972116	2	5
DOCU	3.021	1.215745	1	5
FLEX	3.487	1.506685	1	6
LTEX	2.951	1.251465	1	5
PCAP	2.99	1.221225	1	5
PCON	2.973	1.174585	1	5
PLEX-	3.045	1.235517	1	5
PMAT	3.508	1.503395	1	6
PREC	3.504	1.499411	1	6
PVOL	3.545	0.967942	2	5
RELY	2.897	1.214028	1	5
RUSE	3.969	1.228637	2	6
SCED	2.98	1.216997	1	5
SITE	2.955	1.211793	1	5
STOR	4.571	0.978732	3	6
TEAM	3.556	1.514978	1	6
TIME	4.544	0.95968	3	6
TOOL	2.998	1.245611	1	5
RISK-	6.683	6.373	0	42
EFFORT-	30807.5	33883.81	19	234541

TABLE III: coc1000

	mean	std	min	max
IDX	250	144.1932	1	499
AFP	486.8577	1059.171	9	17518
Input	167.0982	486.3386	0	9404
Output	113.6012	221.2744	0	2455
Enquiry	61.6012	105.4228	0	952
File	91.23447	210.271	0	2955
Interface	24.23447	85.041	0	1572
Added	360.3547	829.8423	0	13580
Changed	85.06212	290.857	0	5193
Deleted	12.35271	124.2241	0	2657
PDR_AFP	11.32265	12.09709	0	149.5

TABLE IV: china

	mean	std	min	max
Loc+	1009.038	574.7463	2	2000
Acap	3.0042	1.209764	1	5
Aexp	2.9911	1.216211	1	5
Arch	3.5159	1.504385	1	6
Cplx	3.5224	1.499174	1	6
Data	3.501	0.960776	2	5
Docu	3.0017	1.226967	1	5
Flex	3.5005	1.498274	1	6
Ltex	3.0024	1.217351	1	5
Pcap	3.0008	1.232295	1	5
Pcon	2.9941	1.224996	1	5
Plex	3.0221	1.223258	1	5
Pmat	3.5188	1.498623	1	6
Prec	3.5263	1.501111	1	6
Pvol	3.5134	0.956824	2	5
Rely	2.9948	1.215039	1	5
Ruse	3.998	1.228229	2	6
Sced	2.9885	1.224385	1	5
Site	3.0025	1.220673	1	5
Stor	4.5059	0.954283	3	6
Team	3.5061	1.503659	1	6
Time	4.5145	0.960043	3	6
Tool	2.998	1.221043	1	5
Risk-	6.5854	6.042165	0	44
Effort-	30506.37	35435.43	3	449147

TABLE V: coc10000

The tables provided above present the statistical description of several datasets in the form of mean, standard deviation, minimum and maximum values of the columns present in each file. The datasets include auto93, auto2, coc1000, china and coc10000.

It is important to note that some of the row headers are denoted with either a '+' or a '-' symbol, which signifies that these variables are the objectives of the dataset. In other words, they are supposed to be either maximized or minimized, depending on the symbol assigned to them.

We are only presenting statistical information for these 5 datasets due to space limitations. However, statistical data for the remaining 6 datasets, namely pom, ssn, ssm, nasa93_dem, healthCloseIsses12mths0001-hard, and healthCloseIsses12mths0001-easy, can be similarly represented but it would be worthy to note that these have the following number of (Xs,Ys) attributes: (10,3), (17,2),(13,2), (22,4),(5,3),(5,3) respectively.

C. Performance

Our performance can be checked against the dataset when sorted via Zitzler's predicate. Taking into account 20 runs of the algorithm, we got the following values as an average of all the runs for each attribute of 'auto93.csv'[3] : [sway2 1997.08 18.05 36.15 8]. If we check this with the sorted dataset, based on Ziztler's domination, our row would fall well within the top 80 rows. It would be ranked 23. This is an indication that our algorithm is competitive enough, if not the best. In another variant of sway which would be explored later, the rank would be 50. The results are further explained in the next section but the performance is tested by comparing it against the sorted dataset.

Budget is always an important constraint and the number of times we access the Y-values decides how costly our process is going to be. If we crack the code of fewer y lookups and somehow manage to achieve closeness to utopia, we have reached success. We observe that the baseline model does N lookups of y where N is the size of the dataset but SWAY could do way better with very few lookups. We can use 'evals' or evaluations as our budget decider. Examining the functioning of SWAY, we learned that the stopping condition is decided either by evals or the number of points that are left in the final cluster. The key decider to this is the hyper-parameter 'the.min' and hence we performed a budget study by trying out with different budgets, i.e, with different values of 'the.min'. The performance of our algorithm varied over different budgets whose results are explained in the next section.

D. Summarization

The results of the budget study would decide which of the variants of SWAY did well when we increased our budget. It may not always be the case that increasing budget would yield us better results. This translates to the concept of overfitting where the model might have high training accuracy but less test accuracy.

Once all the methods have been explored, we need to check for the practical significance of the difference between each of the groups. The results generated by sway2 and xpln2 can be considered significantly different from all and xpln only if we would be able to pass them through a test and check the significance level of the test. Differences although present might not be that significant and this means that they might have arisen due to chance. On the other hand, we also have an effect size test and this deals more with quantifying the magnitude of that difference, Cliff's delta is the effect size test that would be used. Since we are dealing with optimization, there is an inherent concept of ranks and Cliff's delta uses the number of times that one item in the first group is ranked higher than an item in the second group, minus the number of times that an item in the second group is ranked higher than an item in the first group. The distribution of the data is unknown and thus it is wise to use a non-parametric test. This is also a robust test and is not affected by outliers or extreme values in the data.

IV. Result

A. Run with different Y budgets

We noticed that budgets are decided by the number of evals and to reduce or increase the number of evals, we tweak 'the.min' hyper-parameter. This decides the stopping condition and how many points are left in the last resulting cluster. Here are the results with different budgets:

For a value of the.min=0.5, we get:

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	1997.08	18.05	36.15	8
xpln	1985.3	17.231	32.574	8
top	1994.67	19.41	40.83	398

TABLE VI: the.min=0.5

We first tested by increasing the budget, i.e, increasing the evals by reducing the value of the.min to 0.35. We see that:

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	2014.25	17.9	38.75	10
xpln	1985.82	17.27	32.492	10
top	1989.71	20.11	41.43	398

TABLE VII: the.min=0.35

We further increase the budget and set the.min to 0.20. We see that:

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	1973.33	18.63	40	12
xpln	2143.16	17.037	32.0212	12
top	2222.5	20.8	45	398

TABLE VIII: the.min=0.20

We see that as we increased the budget slightly, sway2 did not do so well but when we bumped up the budget even more, it did better than what it initially did. This gives us an indication that when we allow our model to look at more y's it does slightly better.

We now tested the opposite by reducing the budget, i.e, by increasing the.min to 0.62.

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	1994.1	17.18	34.62	6
xpln	2109.75	16.776	31.3995	6
top	2006.03	18.67	36.84	398

TABLE IX: the.min=0.62

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	2059.38	17.12	32.44	5
xpln	2107.09	16.753	31.3195	5
top	2093.51	17.91	34.07	398

TABLE X: the.min=0.75

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	2209.52	16.59	30.12	3
xpln	2227.02	16.45	29.85	3
top	2250.48	16.94	31.65	398

TABLE XI: the.min=0.90

We can observe that as we decrease the budget in the form of increasing the.min value, the model does not get better. This is evident from the fact that under each attribute, for example Lbs- which expects us to minimize lbs, we started off at 1997 when the.min was 0.5 but when we increased the budget to 0.9, we got Lbs value to be 2209 and this shows that sway2 did better initially and not now.

B. Prudence study and further explanation

We also noticed that in 20 runs, there was no randomization at all with our sway2 agglomerative approach. In order to incorporate randomness and then leverage the average metric of the data so obtained from the 20 iterations, we used a random point in the cluster and not the centroids of the cluster as we did in agglomerative clustering. Once the clusters are produced by agglomerative clustering, two random points from each of the clusters are selected and passed to the better function, which decides the better cluster to recursively cluster on. The results obtained by this sway3 is :

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway3	2109.88	17.2505	31.6625	6.65
xpln	2210.84	16.834	30.204	6.65
top	1973.35	19.42	39.913	398

TABLE XII: sway3:Randomization in agglomerative clustering

If we observe the results of sway3 and the results of sway2 got from TABLE VI, we observe that sway2 did better. Now, it was interesting to find out that not only can we stick to agglomerative clustering, but, we can also draw inspiration from sway1 and incorporate some of its workings. Thus we came up with another algorithm for SWAY where we pick two representative points, say A and B from the clusters generated by agglomerative clustering and we draw a line AB. All points in the space are projected onto this line and then we continue with finding the better half and recursing on that. This approach gives us the following results:

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway4	2085.04	17.3845	32.633	6
xpln	2117.83	16.795	30.9495	6
top	1993.15	19.4705	40.874	398

TABLE XIII: sway4

We also talked about the rule generation algorithm. We considered generating all combination of rules and the results of xpln2 when run on SWAY are given below:

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway2	2251.27	16.955	31.735	6
xpln2	2346.07	16.3785	29.096	6
top	1992.19	19.509	40.902	398

TABLE XIV: xpln2 run on best and rest generated by SWAY

We observe that xpln2 also takes the same number of evaluations as SWAY, as it should, since it does not require any extra evaluations once the best and rest rows are generated by SWAY. It uses bins and converts them into rules based on the value function. When xpln2 is compared with xpln, when run on the results of SWAY we see that, it does not do any better than xpln. This means that using the other combination of rules as explained in our ALGO section, was not very useful. This makes us think of the apriori principle [10] which behaves very similarly with rule generation. It states that any subset of a frequent item-set must be frequent which also deduces the notion that an item set when combined with other item sets will only be as much frequent as it was when alone. It cannot increase its support whatsoever when new rules are appended to it.

Overall we see that our implementations on SWAY, i.e, sway2(agglomerative clustering),sway3(agglomerative clustering with random points as centroids) and sway4 (agglomerative + dimensionality reduction) did slightly better than SWAY and it gave competitive results. We can only explain the differences through significance and effect size tests and the results of cliff delta test on Sway4 and xpln gave the following:

method	Lbs-	Acc+	Mpg+
all to all	=	=	=
all to sway4	!=	!=	!=
sway4 and xpln	!=	!=	!=
sway4 and top	!=	!=	!=

TABLE XV: Result of significance test

In the areas which indicate the 'not equal' sign, we can comment that there was a significant difference between the two groups. Ideally, we would want sway4 as close to top as possible and we would want to see an equal-to sign there. We use both significance testing and effect size testing and if the result of both shows statistical difference, only then will our matrix reflect the not-equal-to sign. We used Bootstrap**??** as our significance test while we used Cliff's delta as our effect size test.

method	Lbs-	Acc+	Mpg+	n_evals avg
all	2970.42	15.57	23.84	0
sway	2223.85	16.867	32.0265	6
sway2	1997.08	18.05	36.15	8
sway3	2109.88	17.2505	31.6625	6.65
sway4	2085.04	17.3854	32.633	6
top	1992.19	19.509	40.902	398

TABLE XVI: comparision of all variants of SWAY

Although our implementations Outperformed SWAY for this dataset, it is now a question that which among them rank better. We would not take sway2 to be the best which is going against the data because it has an apparent shortcoming. This algorithm generates the same two clusters every iteration and there is no variance in results at all. Capturing the variance is very important while performing experiments. Even if we run multiple iterations to gather more insights, our algorithm will yield the same thing every time since no seed change is observed. Thus, incorporating randomness and also reducing the dimensionality of data simultaneously, sway4 provides the best result. It reduces the sampling tax since whatever we lost by not looking at all the data is pretty low. We accomplished nearness to the top by using only 6 evaluations.

V. Discussion

Our algorithm or extension of SWAY was successful in performing well for the given set of data but if we have some data that is extremely sparse, our way of tackling the problem may not be valid. In our case, we have a good amount of Xs which help us in semi supervised clustering but if there are certain datasets which have more Ys than useful Xs, our clustering algorithm might not be valid in such cases.

In an attempt to beat SWAY, we learnt about techniques to potentially solve a multi-objective optimization problem and we explored techniques on clustering. In this process we explored four different variants of SWAY and we tried to write our rule generator which was an extension of the existing xpln's functionality. However, we failed to appreciate the time complexity of SWAY which was log linear and we went ahead with trying out polynomial time approaches. We performed hyper-parameter tuning manually, eg: changing 'the.min', but could have always automated the process by using one of the python libraries for grid-search. Moreover, no other hyper-parameters were tweaked and we hence could not attempt to use gradient descent in order to find a global minima for a cost reduction function which involves all the hyper-parameters. We were so inspired by clustering that we did not experiment with non clustering semi-supervised techniques. There are better techniques out there such as weighted sum methods, pareto optimization and evolutionary algorithms, none of which were studied and tried out. Furthermore, to easen out the complexity of our implementation, we also used the Ys while computing our stats and this would lead to questions on validity since the process of computing centroids and similar metrics must not be influenced by the Ys of the data object. Only the arithmetic average of Xs should be taken since any outstanding Y value can influence the metric computation dramatically.

There is always room for improvement and we aim to explore another optimization technique which involves some sort of a cost function. For example, we could give all the attributes equal weight at the beginning and tweak them along the way based on their importance. Moreover, each attribute can be taken as-is or by its inverse value depending on whether we want to maximize or minimize the attribute. We could also explore the same on a more challenging dataset which in today's world would be termed as 'big data'. We might want to run a map-reduce on that sort of data and pass our clustering algorithm to batches of the data and then collect it all using an aggregation function. This would not only be relevant in today's age of growing data, but it would also help come up with a stronger algorithm that could deal with diverse datasets.

VI. Conclusion

The questions that we initially had were all answered with data and experiments. On trying out different algorithms of recursive clustering, it was found out that each one of them performed better than the baseline model. Out of all the models, sway4 performed the best followed by the other two variants of sway called sway2 and sway3. There was a statistically significant difference between sway4 and all, sway4 and xpln, and sway4 and top. Our modifications to the rule-generating algorithm yielded a result that was not as good as the original rule generator. Overall, semi-supervised clustering was explored and studied in depth whilst trying out different versions of the algorithm and it was deemed that this approach to solving multi-objective optimization was a valid one since we could visually see the model improve the attributes in accordance with its goals.

VII. Extra Credit

A. B2: February study

It was learned from January that the agglomerative clustering technique performed badly when there was a reduction in the budget which is evident from Tables IX-XI. However, we like to learn from the doings in January and further improve on them. In this new month, we wish to improve the clustering technique and try to introduce randomness in each iteration. In our earlier case, the clusters were unequally divided and this means that the imbalance in the size would result in irregularities in the number of evaluations. If the smaller cluster wins the Zitzlers domination test, we would see an early stopping since the minimum size is reached early. However, if we recurse on the larger cluster, we would lead ourselves into further evaluations and once we recurse on the larger cluster, another imbalance might creep up which results in uncertainty as to how much more evaluations it would take. Moreover, it would be unfair if the best cluster has lower points since obviously the number of rests would outway it. Thus we need to ensure balance in cluster sizes and introduce randomness in order to fully capture the variability in results. Thus in February, we come up with another technique that uses randomness in a way by selecting two random points in the resulting cluster and this division into clusters is done equally by reducing the dimensionality of data by projecting the points onto the line connecting the two random points. Data is now split into halves and Zitzler's domination is applied to obtain the better half.

B. B3: Ablation Study

In an attempt to perform an ablation study, we follow an approach that closely resembles this process. We start with a model which always chooses the same centroid in every iteration and we gather its performance. We do the same thing but instead of centroids, we turn that feature off and turn on another feature that selects a random representative element in each cluster. This new approach's performance is also gathered. We keep this feature intact but add another feature that helps reduce dimensionality. This algorithm is called sway4 and it was observed to perform the best. This is an incremental approach, which tries to follow an ablation study-like strategy. The same thing was tried with the rules in xpln. We used all the combinations of rules initially, but incrementally dropped each one of the combination of rules. The results for the proposed method of xpln (as learnt in class) worked best.

ACKNOWLEDGMENT

The authors would like to thank Dr. Menzies for his mentorship and guidance. We would also like to thank the Department of Computer Science at North Carolina State University for providing us this opportunity.

REFERENCES

[1] Batool, F. (2019). An agglomerative hierarchical clustering method by optimizing the average silhouette width. ArXiv. /abs/1909.12356
[2] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.
[3] UCI Machine Learning Repository: Auto MPG Data Set. (n.d.). UCI Machine Learning Repository: Auto MPG Data Set. https://archive.ics.uci.edu/ml/datasets/auto+mpg
[4] https://arxiv.org/pdf/1609.05563.pdf
[5] https://arxiv.org/pdf/1608.07617.pdf
[6] Vivek Nair, Tim Menzies, Norbert Siegmund, and Sven Apel. Using bad learners to find good configurations. In Foundations of Software Engineering. ACM, 2017.
[7] ACM Digital Library. (n.d.). ACM Digital Library. https://dl.acm.org/doi/10.1162/106365605774666895
[8] K. Deb, M. Mohan and S. Mishra, "Evaluating the Domination Based Multi-Objective Evolutionary Algorithm for a Quick Computation of Pareto-Optimal Solutions," in Evolutionary Computation, vol. 13, no. 4, pp. 501-525, 2005, doi: 10.1162/106365605774666895.
[9] Menzies, T. (n.d.). Accurate estimates without local data? (PDF) Accurate Estimates Without Local Data? — Tim Menzies - Academia.edu. https://www.academia.edu/2699259/Accurate_estimates_without_local_data
[10] Research on parallelization of Apriori algorithm in association rule mining. (2021, April 19). Research on Parallelization of Apriori Algorithm in Association Rule Mining - ScienceDirect. https://doi.org/10.1016/j.procs.2021.02.109