A Multi-objective Semi-supervised Explanation System and Comparison of Different Sway Methods

Setu Kumar Basak
North Carolina State University, USA
Email: sbasak4@ncsu.edu

***Abstract*—Increasingly, Software Engineering (SE) researchers use optimization techniques to solve SE problems with multiple conflicting objectives. These techniques often require seeing a large number of samples to give better solutions. However, looking into large samples is impractical, costly, and time-intensive. An alternative approach proposed is to start with a very large population and sample down to just the better solutions. The method is called "Sway", short for "the sampling way". However, the present sway method does not work with non-spherical data. *The goal of our paper is to aid researchers in finding better solutions for non-spherical data by looking at a few samples using a novel sway method*. We proposed a novel sway method using the DBSCAN clustering algorithm. We also compared our sway method with another sway method that uses recursive fast-map clustering. We found that our sway method is performing well with less number of evaluations in some cases with respect to the other sway method. We recommend investigating more on tuning the parameters used by our sway and using the method to find better solutions with less budget.**

I. Introduction

Software engineers often need to answer questions exploring trade-offs between competing goals. For example:

- What is the least number of samples to label?
- What is the smallest set of test cases that cover all program branches?
- What is the set of requirements that balances software development cost and customer satisfaction?
- What sequence of refactoring steps takes the least effort while most decreasing the future maintenance costs of a system?

Search-based software engineering (SBSE) is a commonly used technique for solving such problems. Two things are required for using SBSE methods:

- The model is some device that generates multiple outputs (one for each objective) if its inputs are perturbed.
- The optimizer is a device that experiments with different model inputs to improve model outputs.

Different models can require different optimizers. According to Wolpert and Macready [1], no single algorithm can ever be best for all optimization problems. They caution that for every class of problem where algorithm A performs best, there is some other class of problems where A will perform poorly. Hence, when commissioning a new domain, there is always the need for some experimentation to match the particulars of the local model to particular algorithms. When conducting such commissioning experiments, it is very useful to have a baseline optimizer, such as an algorithm that can generate floor performance values. Such baselines let developers quickly rule out any optimization option that falls "below the floor". In this way, researchers and industrial practitioners can achieve fast early results while also gaining some guidance in all their subsequent experimentation (specifically: "try to beat the baseline").

A previous study by Chen et al. [2] proposed a new algorithm called SWAY (short for the sampling way) as a baseline optimizer for search-based SE problems. SWAY has all the properties desirable for a baseline method, such as simplicity of implementation and fast execution times. Further, the experiments of the paper show that SWAY usually performs as well as, or better than, more complex algorithms even for some very hard problems, such as selecting candidate products according to five objectives from highly constrained product lines. Most importantly, SWAY adds very little to the overall effort required to study a new problem. We term that SWAY algorithm as sway1 in our paper. We observe that sway1 uses a recursive fast-map clustering method where the samples are halved in each clustering step. However, for non-spherical data, some of the better samples can reside in cluster2 as sway1 rejects cluster2 and accepts cluster1 based on the overall cluster score using Zitzler's predicate [3]. In addition, two clusters may not have an equal number of samples. One cluster can have more samples, and another cluster can have fewer samples. In addition, the recursive clustering applies to spherical clusters, and its accuracy will suffer if the data is non-spherical. Figure 1 shows how recursive bi-clustering fails to cluster the data into the correct cluster, whereas density-based clustering detects two clusters correctly.

(a) Recursive Clustering

(b) Density-based Clustering

Fig. 1: Clustering of non-spherical data

The goal of our paper is to aid researchers in finding better solutions for non-spherical data by looking at a few samples using a novel sway method.

We hypothesize that applying density-based clustering will remove noise efficiently and provide better solutions with less evaluations. In this study, to resolve the non-spherical data clustering problem, we proposed sway2 using a density-based spatial clustering technique and answered the research question below.

- **RQ:** How does density based clustering (sway2) perform with recursive clustering (sway1)?

We used Density-based spatial clustering of applications with noise (DBSCAN) method for sway2. We observed that DBSCAN clustering in the sway2 method is performing well in some datasets compared to sway2 with less sampling and explanation tax. We list our contributions as follows:

- A novel sway method using a density-based clustering method (DBSCAN)
- Comparison between density-based clustering sway (sway2) and recursive fast-map clustering sway (sway1)

II. Related Works

Accompanied by the rapid development of data acquisition and storage technology, it is easier to obtain unlabeled data. However, it is relatively difficult to obtain labeled samples because of the need to consume a certain amount of manpower and material resources. Semi-supervised learning is a new method between supervised learning and unsupervised learning, whose purpose is to fully use large unlabeled samples to make up for the lack of labeled samples. The main aim of the multi-objective semi-supervised explanation is to study how to use unlabeled samples to help train a supervised learning classifier and only look at small samples to find better solutions when labeled samples are insufficient or burdensome to get by.

Researchers have studied different optimization techniques for multi-objective semi-supervised learning. For example, sampling has been successively applied to noisy real-world optimization problems by Cantu-Paz [4]. They introduced an adaptive sampling policy which they tested on a 100-bit onemax function. While Cantu-Paz demonstrated that adaptive sampling could find better solutions, from our perspective, the drawback with that work is that it requires far more computation time. Shahrzad et al. [5] also analyzed the advantage of age-layering in an evolutionary algorithm. In aged-layered evolutionary algorithms, a small sample of candidates are evaluated first; and if they seem promising, they are evaluated with more samples. The age-layering method effectively reduces the fitness evaluations and speedups the evolution process. However, at least for the aged-layered algorithm reported by Shahrzad et al. [5], this approach still requires millions of model evaluations. Hence, it would not be a candidate for a baseline SBSE method.

Saha et al. [6] proposed a new approach towards semi-supervised learning using the multi-objective optimization (MOO) framework. Four objective functions are optimized using the search capability of a multi-objective simulated annealing-based technique, AMOSA. These objective functions are based on some unsupervised and supervised information. The first three objective functions represent the goodness of partitioning in terms of Euclidean distance, total symmetry present in the clusters, and cluster connectedness. For the last objective function, they considered different external cluster validity indices, including adjusted rand index, rand index, a newly developed min-max distance-based MMI index, NMMI index, and Minkowski Score. Their result showed that the proposed semi-supervised clustering technique could effectively detect the appropriate number of clusters as well as the appropriate partitioning from the data sets having either well-separated clusters of any shape or symmetrical clusters with or without overlaps. However, this approach requires multiple model evaluations as well.

Chen et al. [2] introduced a baseline method, SWAY, to explore optimization problems in the context of search-based software engineering problems. SWAY can find promising individuals among a large set of candidates using a very small number of model evaluations. Since the number of required model evaluations is much less than that of other algorithms, SWAY terminates very early. SWAY would be especially useful when the model evaluation is computationally expensive (i.e., very slow). SWAY satisfies all the criteria of a baseline method, such as simple to code, applicable to a wide range of models. They tested SWAY via numerous scenarios within three SE models. These models differed in their type of decisions as well as the size of their decision space. Their results showed that the quality of outputs from SWAY was comparable to the state-of-the-art evolutionary algorithms for those specific problems. SWAY is also very fast. Among 15 cases studied in their paper, SWAY only requires less than 5 percent (in median) of the runtime of the standard algorithms, such as evolutionary algorithms. However, while quickly clustering the data, SWAY did not account for non-spherical data as it halves the data in each cluster (recursive clustering) and rejects one cluster based on the pivot centroid. As a result, a better sample can be present in the rejected cluster if the clustering is not done properly. Hence, we look to improve the clustering approach of SWAY by using density-based clustering to sample better solutions quickly and correctly. Density-based clustering will also lessen the number of evaluations as we will only check the centroids of each cluster which can be less than the recursive clustering.

III. Methodology

In this section, we provide a) a description of our new sway method, b) the details of the datasets used in this study, and c) comparing our sway method with state-of-the-art sway method.

A. SWAY Method

For sway2, we used the Density-based spatial clustering of applications with noise (DBSCAN) as part of clustering the data. DBSCAN is a data clustering algorithm proposed by Martin Ester et al. [7] in 1996. It is a density-based clustering non-parametric algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away). DBSCAN algorithm basically requires two parameters:

- **Epsilon(eps):** *eps* (ϵ) specifies how close points should be to each other to be considered a part of a cluster. It means that if the distance between two points is lower or equal to this value (eps), these points are considered to be neighbors.
- **minPoints:** *minPoints* refers the minimum number of points to form a dense region. For example, if we set the minPoints parameter as 5, then we need at least 5 points to form a dense region. It's basically known as min_samples.

In DBSCAN algorithm, there are three types of data points. Every point in the dataset D = $\{xi\}$, on given minPoints and eps, we can categorize every data point into Core point, Border point, and Noise point. These points are described below and shown in Figure 2.

- **Core Point:** A point is a core point if it has more than a specified number of minPoints within eps radius around it. Core Point always belongs in a dense region. For example, if we consider 'p' is set to be a core point if 'p' has $\geq$ minPoints in an eps radius around it.
- **Border Point:** A point is a border point if it has fewer than minPoints within eps, but is in the neighborhood of a core point. For example, 'p' is set to be a border point if 'p' is not a core point. i.e 'p' has $>$ minPoints in eps radius. But 'p' should belong to the neighborhood 'q' where 'q' is a core point (p $\in$ neighborhood of q and distance(p,q) $\geq$ eps)
- **Noise Point (Outlier):** A noise point is any point that is not a core point or a border point.

DBSCAN algorithm uses 'Density Edge' and 'Density Connected Points' to build clusters. These concepts are described below and shown in Figure 3.

- **Density Edge:** If p and q both are core points and distance between (p,q) $\geq$ eps then we can connect p, q vertex in a graph and call it "Density Edge".
- **Density Connected Points:** Two points p and q are said to be density connected points if both p and q are core points and

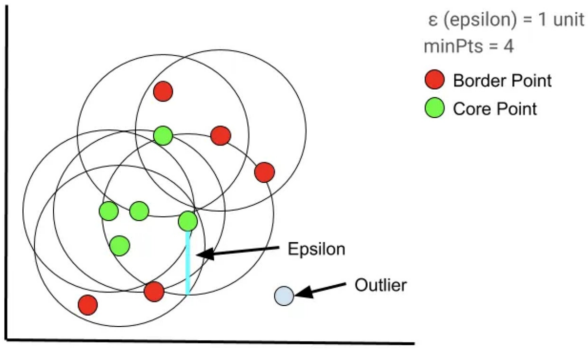


Fig. 2: Depiction of Core point, Border Point and Noise point (Outlier) in DBSCAN with eps and minPoints parameters.

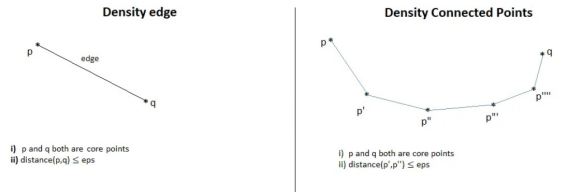


Fig. 3: Depiction of Density Edge and Density Connected Points

they exist a path formed by density edges connecting point p to point q.

Below, we describe the DBSCAN algorithm for finding clusters:

- **Step 1:** $\forall xi \in D$, label points as the core, border, and noise points.
- **Step 2:** Remove or eliminate all noise points. (because they belong to the sparse region. i.e they are not belonging to any clusters.)
- **Step 3:** For every core point p that has not yet been assigned to a cluster, create a new cluster with the point p and add all the points that are density-connected to p.
- **Step 4:** Assign each border points to the cluster of the closest core point.

After finding the clusters using DBSCAN, we applied the following procedures for finding best samples.

- **Finding Centroid:** Since clusters are formed by the process of linking neighbor points together, DBSCAN does not provide any centroid of a cluster. However, we need to identify a representative point of each cluster to find the best cluster among all the clusters found by DBSCAN. To identify a representative point, we took the mid of each column in the dataset for the specific cluster data points and created a mid point. Then, we find the Euclidean distance between the mid point and each point in the cluster. We sorted the points based on the distance and picked the point as centroid, which is closer to the mid point.
- **Finding Best Cluster:** After finding each representative point for each cluster, we use Zitzler's domination predicate to find the better cluster.

B. Xpln2

In this study, we used the xpln1 implemented in our homework as xpln2. The algorithm is described below:

- **Step 1:** Find the ranges using Entropy Merge Discretization (EMD) that distinguish the best cluster from some random sample of the rest.
- **Step 2:** Explore the found ranges looking for rules that select for the best cluster.

Our goal is to reduce the sampling tax and explanation tax.

- **Sampling Tax:** Sampling tax comes since the less we look, the more we miss important stuff. We can calculate the sampling tax by comparing (a) what happens what we get with SWAY and (b) what we get after evaluating all examples.
- **Explanation Tax:** Since prediction is hard and the simplifications we use for explanation compromise predictive performance. We can calculate that by comparing (a) what happens what we get with SWAY and (b) what we get after building and applying some explanation.
- **Explanation Variance:** Explanations generated from a few random probes of a complex multi-dimensional can be widely variable. We calculate the variance by running our explanation algorithm 20 times with different random number seeds.

C. Dataset

In this study, we have used 11 datasets. Table III shows a brief overview of the datasets. The "Dataset Name", "Domain", "Goals" and "Data Count" columns present the dataset name, the dataset's domain in SE, the goals we are trying to minimize and maximize, and the number of data points in the dataset, respectively. The '+' and '-' symbols after the goal columns refer to the maximization and minimization of a specific goal. Below we describe in detail two datasets among 11 datasets.

auto93: The Auto MPG dataset (auto93) is a popular dataset used in the field of machine learning and data mining. It contains information about various cars, including their miles per gallon (mpg) fuel efficiency and other attributes such as the car's horsepower, displacement, weight, and acceleration. Table I shows the overview of the auto93 dataset.

TABLE I: Overview of auto93 dataset

	Clndrs	Volume	Lbs-	Acc+	Model	origin	Mpg+
mean	5.45	193.43	2970.42	15.57	76.01	1.57	23.84
std	1.70	104.27	846.84	2.76	3.70	0.80	8.34
min	3.00	68.00	1613.00	8.00	70.00	1.00	10.00
25%	4.00	104.25	2223.75	13.83	73.00	1.00	20.00
50%	4.00	148.50	2803.50	15.50	76.00	1.00	20.00
75%	8.00	262.00	3608.00	17.18	79.00	2.00	30.00
max	8.00	455.00	5140.00	24.80	82.00	3.00	50.00

coc1000: The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W. Boehm. The model parameters are derived from fitting a regression formula using data from historical projects (63 projects for COCOMO 81 and 163 projects for COCOMO II). The datased used in our study consists of data related to development flexibility, process maturity, precedentedness, and architecture or risk resolution. Table II depicts an overview of the coc1000 dataset.

D. Performance Measure and Summarization:

For performance measures, we used the below steps for each dataset.

TABLE II: Overview of coc1000 dataset

	ACAP	ARCH	CPLX	DATA	DOCU	FLEX	LTEX	PCAP	PCON	PMAT	PREC	PVOL	RELY	RUSE	SCED	SITE	STOR	TEAM	TIME	TOOL
mean	3.07	3.56	3.55	3.46	3.02	3.49	2.95	2.99	2.97	3.51	3.50	3.54	2.90	3.97	2.98	2.96	4.57	3.56	4.54	3.00
std	1.25	1.48	1.47	0.97	1.22	1.51	1.25	1.22	1.17	1.50	1.50	0.97	1.21	1.23	1.22	1.21	0.98	1.51	0.96	1.25
min	1.00	1.00	1.00	2.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	2.00	1.00	2.00	1.00	1.00	3.00	1.00	3.00	1.00
25%	2.00	2.00	2.00	3.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	3.00	2.00	3.00	2.00	2.00	4.00	2.00	4.00	2.00
50%	3.00	4.00	4.00	3.00	3.00	3.00	3.00	3.00	3.00	4.00	3.00	4.00	3.00	4.00	3.00	3.00	5.00	4.00	5.00	3.00
75%	4.00	5.00	5.00	4.00	4.00	5.00	4.00	4.00	4.00	5.00	5.00	4.00	4.00	5.00	4.00	4.00	5.00	5.00	5.00	4.00
max	5.00	6.00	6.00	5.00	5.00	6.00	5.00	5.00	5.00	6.00	6.00	5.00	5.00	6.00	5.00	5.00	6.00	6.00	6.00	5.00

TABLE III: Overview of the Studied Datasets

Dataset Name	Domain	Goals	Data Count
auto2	Car design	CityMPG+, HighwayMPG+, Weight- and Class-	93
auto93	Car design	Lbs-, Acc+ and Mpg+	398
china	Software project estimation	N_effort-	499
coc1000	Software project estimation	LOC+, AEXP-, PLEX-, RISK- and EFFORT-	1,000
coc10000	Software project estimation	LOC+, RISK- and EFFORT-	10,000
nasa93dem	Software effort and defects estimation	Kloc+, Effort-, Defects- and Months-	93
healthCloseIsses12mths0001-hard	Issue close time	MRE-, ACC+ and PRED40+	10,000
healthCloseIsses12mths0011-easy	Issue close time	MRE-, ACC+ and PRED40+	10,000
pom	Agile project management	Cost- and Idle-	10000
SSM	Computational physics	NUMBERITERATIONS- and TIMETOSOLUTION-	239,360
SSN	Computational physics	PSNR- and Energy-	53,662

- **Iterations:** We repeated the steps for 20 iterations with different random number seeds.
- **Limited budget** We selected a limited budget to see the number of goal samples. For smaller datasets such as auto2 and auto93, we have used budgets such as 2, 4, 6, 8, and 10, whereas, for larger datasets such as coc1000, we have used 5,10, and 20.
- **Evaluate top guesses:** We evaluated the top guesses using Zitzler's domination predicate and ranked the solutions.
- **Collect distributions:** We collected the distributions of the better samples after running 20 iterations for a dataset.

To find which method is better, we did statistical tests such as the significance test and the effect size of the distribution of the results. Below we discuss the significance test and effect size used in our study.

Significance Test: The significance test refers to how much one distribution overlaps the other distribution. If a result is statistically significant, it is unlikely to be explained solely by chance or random factors. In other words, a statistically significant result has a very low chance of occurring if there were no true effect in a research study. In our study, we have used bootstrapping [8], a form of hypothesis testing that involves resampling a single data set to create a multitude of simulated samples. Those samples are used to calculate standard errors, confidence intervals, and hypothesis testing. We have selected this non-parametric significance testing as our datasets do not have Gaussian distribution. We have run the significance testing with (`B = 512`) resamples and checked with 95% confidence interval (`p = 0.05`).

Effect Size: Effect size test refers if two distributions are different by a trivial amount. Effect size indicates the practical significance of a research outcome. A large effect size means a research finding has practical significance, while a small effect size indicates limited practical applications. While statistical significance shows that an effect exists in a study, practical significance shows that the effect is large enough to be meaningful in the real world. Statistical significance alone can be misleading because the sample size influences it. Increasing the sample size makes it more likely to find a statistically significant effect, no matter how small the effect is in the real world. In contrast, effect sizes are independent of the sample size. Only the data is used to calculate effect sizes. In our study, we have used a non-parametric effect size test cliff's delta [9]. We have used the cliff's delta threshold of 0.147 (small).

IV. Results

In this section, we discuss the results and findings of our study.

A. Comparison of sway2 with sway1

We ran sway2 and sway1 with different budgets. Table IV and Table V show the result of each dataset against different methods such as sway1 and sway2. Below observations can be made based on Table IV and Table IV.

- **Better:** Sway2 has performed better than sway1 for some datasets. For example, in the auto2 dataset (Table IVa), the 'CityMPG+', 'HighwayMPG+', 'Weight-' and 'Class-' are 42.8, 46.6, 1758.5, and 8.3, respectively for sway1. However, for sway2, the 'CityMPG+', 'HighwayMPG+', 'Weight-', and 'Class-' are 46.0, 50.0, 1695.0, and 8.4, respectively. We observed that we got better samples for 'CityMPG+', 'HighwayMPG+', and 'Weight-'. Similarly, for the coc10000 dataset (Table IVd), the 'LOC+' and 'EFFORT-' are 1818.9 and 17231.6, respectively, for sway1, whereas that of for sway2 are 1974.1 and 13607.5, respectively. We can observe that sway2 has found maximized LOC and minimized EFFORT. Similarly, sway2 has performed better than sway1 in china (Table IVg) dataset.
- **Similar:** We observe that in some dataset sway2 has performed almost similar to sway1. For example, in the pom dataset (Table Vb), the 'Cost-', 'Completion+' and 'Idle' are 128.1, 0.9 and 0.1, respectively for sway1, whereas that of for sway2

	CityMPG+	HighwayMPG+	Weight-	Class-
all	21.0	28.0	3040.0	17.7
sway1	42.8	46.6	1758.5	8.3
xpln1	29.5	35.25	1871.2	9.02
sway2	46.0	50.0	1695.0	8.4
xpln2	30.0	36.0	1695.0	8.4
top	46.0	50.0	1695.0	8.4
all to all	=	=	=	=
all to sway1	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠
sway1 to sway2	≠	≠	≠	≠
sway1 to xpln1	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠
sway1 to top	≠	≠	≠	≠

(a) auto2

	Lbs-	Acc+	Mpg+
all	2800.0	15.5	20.0
sway1	1900.5	19.1	37.5
xpln1	2169.8	16.7	29.5
sway2	2110.0	17.9	50.0
xpln2	2155.0	16.4	30.0
top	2130.0	24.6	40.0
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠

(b) auto93

	LOC+	AEXP-	PLEX-	RISK-	EFFORT-
all	1067.1	3.0	3.0	5.1	19421.6
sway1	1456.0	2.1	1.5	3.4	26419.0
xpln1	1061.7	2.9	3.0	4.7	19611.4
sway2	1037.0	2.0	1.0	4.0	24987.0
xpln2	1046.7	3.0	3.0	4.6	20549.0
top	1942.0	1.0	2.0	3.0	31323.0
all to all	=	=	=	=	=
all to sway1	≠	≠	≠	≠	≠
all to sway2	≠	≠	≠	≠	≠
sway1 to sway2	≠	≠	≠	≠	≠
sway1 to xpln1	≠	≠	≠	≠	≠
sway2 to xpln2	≠	≠	≠	≠	≠
sway1 to top	≠	≠	≠	≠	≠
sway2 to top	≠	≠	≠	≠	≠

(c) coc1000

	LOC+	RISK-	EFFORT-
all	1006.0	5.1	20053.3
sway1	1818.9	0.1	17231.6
xpln1	101.0	4.4	18464.5
sway2	1974.1	0.3	13607.5
xpln2	1008.4	5.3	20233.1
top	1973.1	0.3	13025.3
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠
sway2 to top	≠	≠	≠

(d) coc10000

	MRE-	ACC+	PRED40+
all	119.3	-12.2	0
sway1	0	0	83.3
xpln1	119.1	-12.1	0
sway2	0	0	83.3
xpln2	107.4	-10.1	6.9
top	0	0	83.3
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	=
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	=
sway2 to top	≠	≠	=

(e) healteasy

	MRE-	ACC+	PRED40+
all	75.0	7.1	25.0
sway1	70.6	9.0	22.2
xpln1	74.6	7.2	25.0
sway2	71.2	22.8	31.9
xpln2	75.4	7.0	25.0
top	64.7	25.6	31.9
all to all	=	=	=
all to sway1	≠	≠	≠
all to sway2	≠	≠	≠
sway1 to sway2	≠	≠	≠
sway1 to xpln1	≠	≠	≠
sway2 to xpln2	≠	≠	≠
sway1 to top	≠	≠	≠
sway2 to top	≠	≠	≠

(f) healthhard

	N_effort-
all	2098.0
sway1	320.5
xpln1	938.0
sway2	140.0
xpln2	759.9
top	31.0
all to all	=
all to sway1	≠
all to sway2	≠
sway1 to sway2	≠
sway1 to xpln1	≠
sway2 to xpln2	≠
sway1 to top	≠

(g) china

TABLE IV: Comparison of sway2 with sway1

are '136.0', '0.9', and '0.1', respectively. We can observe that the Completion and Idle results are same for both sway1 and sway2.

- **Not Better:** We observe that sway2 has under performed than sway1 in datasets such as nasa93dem (Table Va). The 'Kloc+', 'Effort', 'Defects', and 'Months' are 8.5, 38.3, 379.3, and 11.0, respectively, for sway1, whereas that of sway2 are 6.2, 43.4, 352.4, and 11.8, respectively. We can observe that sway2 performed worse than sway1 in both the maximization and minimization of the goals.

B. Prudence Study

We performed a prudence study to check if sway2 performs better when the budget increases. We ran the sway2 method with budget (B = 5, 10, and 20) on the coc10000 dataset. As shown in Figure 5, sway2 found a better sample with more minimized 'Effort' when the budget is increased. At budget (B = 5), the value of 'Effort' is 20786, and that of at B = 20 is 18928.

C. Requirements Study

We chose Netflix movies as the topic for the requirements study. There are 10 attributes and 10 examples in our requirements study which are shown in Table VI. We ran the recursive bi-clustering of the data and compared it against the results of other humans. We observe that two out of five humans has matched the recursive clustering. In the cluster, we observe similarities between attributes such as (WatchOnceMovie:WatchMultipleMovie) and (BadMovie:GoodMovie). The relation between these attributes is obvious as users will not watch a bad movie multiple times, whereas users will watch a good movie multiple times. In addition, we have observed a similarity between (NonExpensiveMovie:ExpensiveMovie) and (ClassicMovie:ThrillerMovie). However, this relation may not hold for large data as classic movies can also be expensive, and a thriller movie can be non-expensive. We believe this relation will not hold when we provide a larger dataset. Hence, we can find that the views are not always the same between humans and data.

	Kloc+	Effort-	Defects-	Months-
all	47.5	252.0	2007.0	21.4
sway1	8.5	38.8	379.3	11.0
xpln1	35.4	256.6	1599.5	19.7
sway2	6.2	43.4	352.4	11.8
xpln2	33.0	227.4	1620.2	20.3
top	0.9	8.4	28.0	4.9
all to all	=	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$	$\neq$

(a) nasa93dem

	Cost-	Completion+	Idle-
all	323.1	0.8	0.2
sway1	128.1	0.9	0.1
xpln1	346.6	0.8	0.2
sway2	136.0	0.9	0.1
xpln2	327.1	0.8	0.2
top	115.9	0.9	0.0
all to all	=	=	=
all to sway1	$\neq$	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$	$\neq$
sway1 to sway2	$\neq$	=	=
sway1 to xpln1	$\neq$	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$	$\neq$

(b) pom

	NUMBERITERATIONS-	TIMETOSOLUTION-
all	6.8	135.2
sway1	3.8	84.6
xpln1	6.4	130.5
sway2	4.5	68.5
xpln2	6.6	134.3
top	3.5	55.8
all to all	=	=
all to sway1	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$

(c) SSM

	PSNR-	Energy-
all	45.7	1244.8
sway1	28.2	420.5
xpln1	45.4	1524.1
sway2	29.4	397.6
xpln2	45.1	1101.2
top	22.0	342.9
all to all	=	=
all to sway1	$\neq$	$\neq$
all to sway2	$\neq$	$\neq$
sway1 to sway2	$\neq$	$\neq$
sway1 to xpln1	$\neq$	$\neq$
sway2 to xpln2	$\neq$	$\neq$
sway1 to top	$\neq$	$\neq$

(d) SSN

TABLE V: Comparison of sway2 with sway1

TABLE VI: Attributes and Examples of Netflix Repertory Grid

Tags	**Keywords**
(ChildrenMovie: AdultMovie), (ClassicMovie:ThrillerMovie), (AnimeMovie:NonAnimeMovie), (ComedyMovie:HorrorMovie), (WarMovie:RomanticMovie), (ShortMovie:LongMovie), (WatchOnceMovie:WatchMultipleMovie), (BadMovie:GoodMovie), (NonExpensiveMovie:ExpensiveMovie), (EducationalMovie:EntertainerMovie)	RushHour, Sully, BicycleThieves, Parasite, AmericanSniper, SocialNetwork, LordOfTheRings, KingsMan, Titanic, IceAge

D. February Study

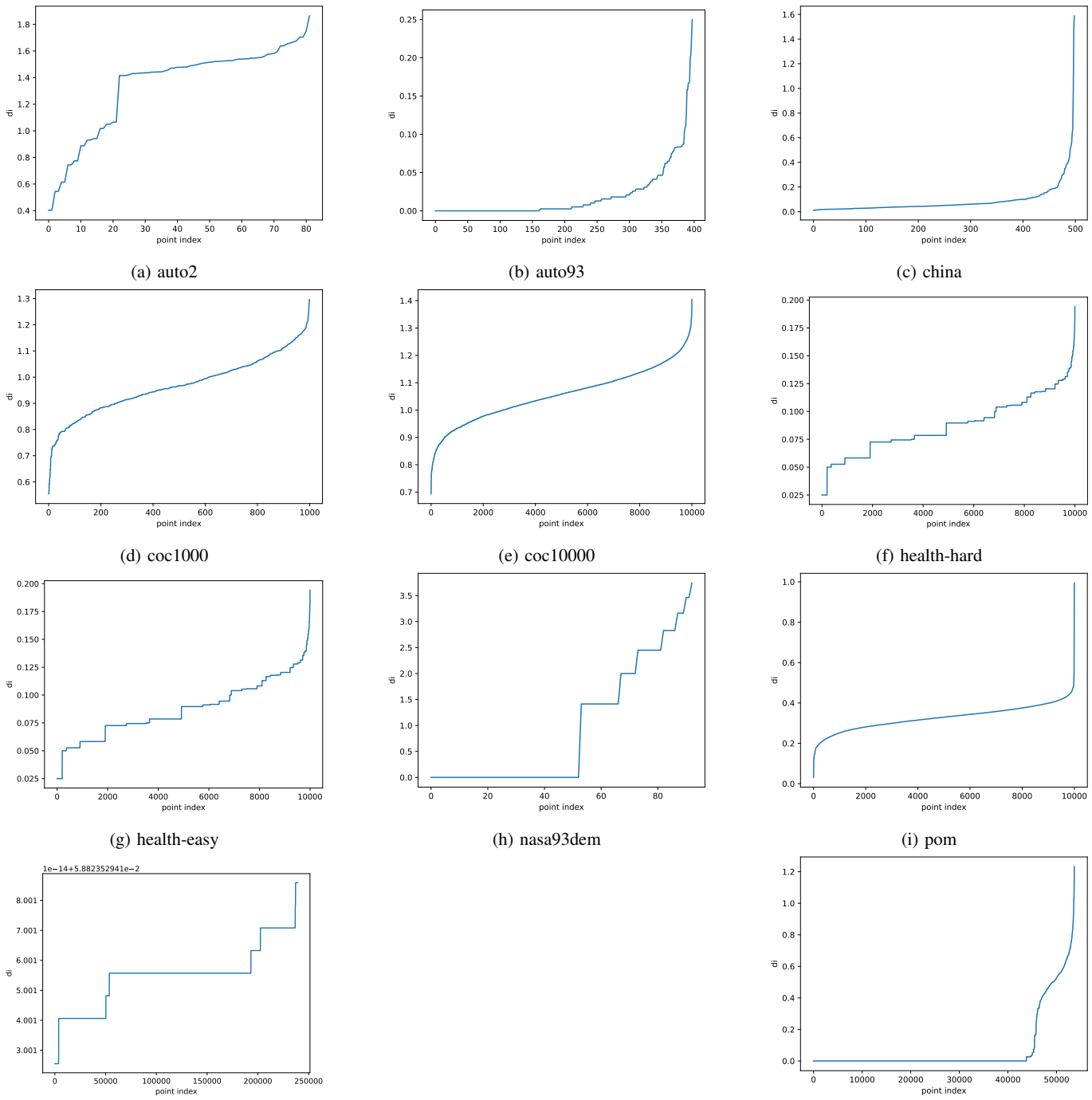
February study refers to a study if the same problem which is solved in January can be solved in February with fewer budgets. By reviewing the budget and results from the January analysis, analysts can identify areas where they may have overspent or where they can optimize their spending. This information can help them allocate their budget more effectively and avoid unnecessary expenses in the future. Additionally, they may have gained insights into the data or methods used in the January analysis, which can help them streamline their process and achieve similar results with less budget. Overall, applying learnings from past analyses can lead to a more streamlined and cost-effective approach to future analyses. For example, in the case of the sway2 method, at first, we used all the data for clustering using the DBSCAN algorithm and used higher budgets. However, we observed from the data that many similar samples are present in the data, and when we look at B samples of our budget, there are some similar-looking data. So, motivated by this information, we selected a random sample of the dataset and lowered the budget, and we got the same result in February.

E. Ablation Study

DBSCAN clustering algorithm in our sway2 method outputs the found clusters and the noises. We selected the rest samples from the rejected clusters and discarded the noise samples. However, in our ablation study, we tried to choose the rest samples, both from the noise and the rejected cluster samples. We observe that after choosing the rest sample from noise, we are not finding rules for the explanation system, and if any rule is found, the explanation tax is high. For example, we ran the algorithm by selecting rest samples with and without noise for the coc10000 dataset. Figure 6 shows that the explanation tax increased when we selected the rest sample from noise. So, it is evident that selected rest samples from the rejected clusters is an important aspect in sway2 method.

F. HPO Study

One important hyper-parameter for the DBSCAN clustering algorithm is epsilon (ϵ). If ϵ is chosen much too small, a large part of the data will not be clustered, whereas for a too-high value of ϵ, clusters will merge, and the majority of objects will be in the same cluster. Hence, to find the optimum ϵ value, we have chosen ϵ by using a k-distance graph, plotting the distance to the $k = minPts - 1$ nearest neighbor ordered from the largest to the smallest value. The minPts are selected based on the number of columns of a dataset under study. The aim is to determine the "knee" corresponding to the optimal epsilon parameter. A knee corresponds to a threshold where a sharp change occurs along the k-distance curve. Figure 4 shows each dataset's optimal (ϵ) range. For example, the optimal epsilon range for auto93 (Figure 4b) and china (Figure 4c) datasets are 0.02 - 0.15 and 0.15 - 0.4, respectively. As we found the optimal range of epsilon for each dataset, further improvement can be made



(a) auto2 (b) auto93 (c) china

(d) coc1000 (e) coc10000 (f) health-hard

(g) health-easy (h) nasa93dem (i) pom

(j) SSM (k) SSN

Fig. 4: HPO study of epsilon values in each dataset of our study.

by running our sway2 method for each value of the epsilon range and checking which exact epsilon value performs the best.

V. Threats to Validity

In this section, we briefly discuss the limitations of our paper.

- **Running Time:** Since we used the DBSCAN clustering algorithm for our sway2 method, each point is checked with the neighboring points within the epsilon distance to make a cluster. We applied Breadth-first search to apply the DBSCAN clustering algorithm. As a result, the running time is very high compared to sway1. Future research should be conducted to improve the running time of the DBSCAN algorithm. However, to mitigate the running time challenge, we have taken a random sample of the larger datasets before applying the DBSCAN algorithm. We got samples with very less sampling tax.
- **Effectiveness:** Our sway method has not performed better for all datasets. However, it could sample better samples with less budget for some datasets such as auto2 and china datasets. Future research can be conducted to find out the reason of not performing better in other datasets by checking the data distribution and tuning the hyper-parameters of sway2 method.
- **Requirements Study:** For requirements study, we could get

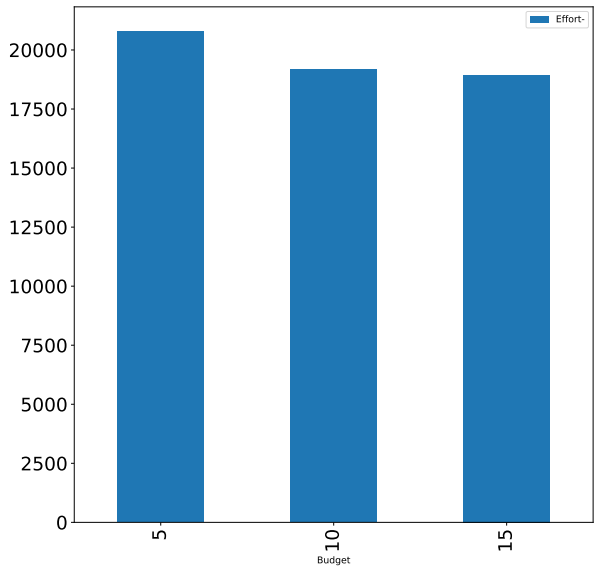


Fig. 5: Better sample of Effort with the increasing budgets for coc10000 dataset.

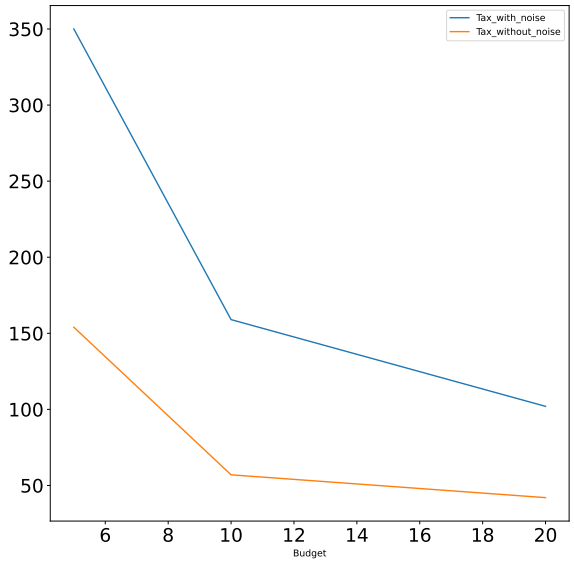


Fig. 6: Ablation study for Loc+ of coc10000 dataset.

three people on board to conduct the repertory grid. However, to avoid any bias, we selected the three people based on ethnicity, culture and age.

VI. Conclusion

We provide a novel sway method (sway2) using the density-based spatial clustering DBSCAN algorithm. Our sway2 finds better samples with fewer evaluations in the datasets by clustering related neighbor points in a big cluster. Though the sway1 method performed better for some datasets, sway2 with DBSCAN clustering can be used to get quick results without looking at all the data. We invite research communities to expand the findings of the sway2 method and make the method more robust by solving the limitations.

References

[1] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[2] J. Chen, V. Nair, R. Krishna, and T. Menzies, ""sampling" as a baseline optimizer for search-based software engineering," *IEEE Transactions on Software Engineering*, vol. 45, no. 6, pp. 597–614, 2019.

[3] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[4] E. Cantú-Paz, "Adaptive sampling for noisy problems," in *Genetic and Evolutionary Computation – GECCO 2004*, K. Deb, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 947–958.

[5] H. Shahrzad, B. Hodjat, and R. Miikkulainen, "Estimating the advantage of age-layering in evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2016, Denver, CO)*, 2016. [Online]. Available: http://nn.cs.utexas.edu/?shahrzad:gecco16

[6] S. Saha, A. Ekbal, and A. K. Alok, "Semi-supervised clustering using multiobjective optimization," in *2012 12th International Conference on Hybrid Intelligent Systems (HIS)*, 2012, pp. 360–365.

[7] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[8] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[9] N. Cliff, "Dominance statistics: Ordinal analyses to answer ordinal questions." *Psychological bulletin*, vol. 114, no. 3, p. 494, 1993.