Bare Demo of IEEEtran.cls for IEEE Conferences

Elizabeth Lin
Department of Computer Science
North Carolina State University
Raleigh, North Carolina
Email: etlin@ncsu.edu

***Abstract*—The abstract goes here.**

I. Introduction

This demo file is intended to serve as a “starter file” for IEEE conference papers produced under LaTeX using IEEEtran.cls version 1.8b and later. I wish you the best of success.

mds

August 26, 2015

A. Subsection Heading Here

Subsection text here.

1) Subsubsection Heading Here: Subsubsection text here.

II. Related Work

The algorithm used in this paper, Sway [1], is a method for search-based software engineering. Search based software engineering (SBSE) was first proposed by Harman and Jones [2] in 2001. SBSE transforms a software engineering problem to a search problem to apply metaheuristic search. A software engineering problem can be reformed as a search problem by defining the following: a representation of the problem, a fitness function, and a set of manipulation operators. Common algorithms include random search, simulated annealing, genetic algorithms.

The advantage of metaheuristic algorithms is that they can explore multiple objectives simultaneously. Multiple-objective evolutionary algorithms (MOEA) are used in SBSE to help achieve multi-objective optimization (MOO). In multi-objective problems, there usually isn’t a single optimal solution [3], a set of optimal points, such as the pareto frontier, is determined.

A. Random Projection

B. Semi-Supervised Learning

C. Why Heuristics Work

III. Methods

Sway recursively splits the dataset in half and finds the best cluster using a *split* function. The *split* function picks a random point and the two points with the largest Euclidean distance from it. The two furthest points form a line, which split all other points into two halves by calculating the distance of the x columns (the columns that are not objectives). Then one point from each of the two halves are compared through the *better* function, to determine which point has better objectives (y columns). The *split* function is then executed on the better half. This process repeats until we reach a certain cluster size, by then we would have the best cluster.

Our new proposed method built on top of sway is to add randomness when splitting data into two halves. The *half()* splits data into two halves in sway, it first sorts all data according to the x columns. These x columns provide a basis for us to cluster similar data together. We do not compare the y columns when splitting data since we assume that in real-world scenarios it would be costly to acquire such values. However, we do don’t know the exact relation between the x columns (characteristics) and the y columns (objectives). Thus, we propose adding randomness, so we don’t fully rely on unknown patterns in the data. We hope that the introduced randomness will help us find better objective values. In the following sections, we refer to the original sway as sway1, and sway2 as our new method.

We next describe the main implementation of randomness in our code. The `dist` function in the data class calculates the distance of the x columns for two rows. We use a random coefficient (`RAND` in the pseudocode below), to vary the distance by a certain percentage. In our experiments, we set this percentage to 15%. This means a distance of 1 would return a value ranging from -0.85 to 1.15.

```
def dist(row1, row2):
    n = 0
    d = 0
for col in x_columns:
    n += 1
    d += col.dist(row1[col_index],
                row2[col_index])**dist_coefficient
    d = d * (1+ random_num(-RAND, RAND))

    return (d/n)**(1/dist_coefficient)
```

We then applied sway to the *xpln* method. The *xpln* method takes the best cluster from sway and another random sample of data and tries to find a rule that would most effectively distinguish between the two. Xpln1 would be sway1 applied to xpln, and xpln2 is sway2 applied to xpln.

The *top* method goes through all data in the dataset to compare and sort based on a *better* metric. The *better* metric compares the y columns of the data. As we compared each pair of data points, this resulted in $O(n^2)$ time. As some datasets had a large number of rows, we did not run the *top* method for certain datasets due to time concerns.

A. Data

Our data consists of ten datasets, each has two types of columns. Columns with string values and columns with numeric values. Some columns have a plus or minus sign, this means they are the objectives, a minus sign means we try to minimize this value, while a plus sign means we try to maximize the value. Table I shows statistics on these objective columns.

For example, the auto2 dataset has a total of twenty-three columns, which includes four objectives, or y columns: *CityMPG+, HighwayMPG+, Weight-, Class-*. The remaining nineteen columns, *type, engine_size, horsepower, etc.*, are x columns. Our methods would cluster data based on the nineteen x columns to maximize or minimize the values of the objectives.

B. Experiments

We ran our model on samples of size 10, 25, 50, 100, 200, 500, and 1000. For each sample size, we ran 20 repeated runs with different seeds and calculated the average of the values we collected. The source code and experiment outputs can be found in the GitHub repository.

IV. Results

In this section, we discuss results of our experiments. We present our results from a sample size of 500 out of the many sample sizes. We observe that adding randomness in our model achieved mixed results. In most cases, sway2 performs slightly worse or similarly to sway1. However, there are certain cases where sway2 performs better. For example, sway2 performs better when we try to maximize *Kloc* for the *nasa93dem* dataset. We indicate objectives where sway2 performs better than as red in our tables. As sway2 introduces randomness, we can infer that the characteristics in the dataset may have a smaller correlation to the objective values, or that closely clustered x columns do not have as strong a correlation to certain objectives. For example, closely clustered x columns for coc1000 may not have similar values for the *Effort* objective.

A. Sample size

We also conducted experiments with different sample sizes in an attempt to achieve better results with more samples. However, we did not observe a clear correlation between sample sizes and the objectives. Table XII shows varying sample sizes for the *nasa93dem* dataset. Upon receiving these results, we went back to check how sample size was implemented in the code. Sample size has an effect when selecting the two points when splitting data into two halves. The first point is selected at random, and the second point is selected from a group of data that is of sample size. However, that is the extent of which sample size has an effect. Thus, we infer that sample size does not have enough effect in our code to influence the results.

B. Effect size and Significance tests

With results from different models, we conducted the effect size test using cliff's delta and the significance test using bootstrap. The effect size test tests for relationship between two sets of data. We set the threshold to 0.4 for medium effects. The confidence for the significance test is set to 0.05. The conjunction of the two test between different results is shown in table XIII. For those where the conjuction is =, this means both tests return true, which we can infer to meaning the two results are similar. We observe that results from sway1 is similar to results from sway2. However, both results from sway are not similar to results from top. It can also be seen from the table that xpln returns slightly different results from sway.

V. Discussion

As sway is the current state-of-the-art method, we spent the majority of our time unpacking and understanding sway. We did not want to completely reinvent the wheel, thus our new method builds on top of it. Future work can be done on further extending and making changes to sway to achieve better performance.

A. Threats to validity

Our method was based off of sway, we were given sample code of sway and had to implement it in our own code. While we made the best effort to implement it, there might be small details where we missed or misunderstood. This could result in our experiments and results being slightly inaccurate.

VI. Bonus: Requirements Study

VII. Bonus: February Study

VIII. Bonus: Ablation Study

IX. Bonus: HPO study

References

[1] J. Chen, V. Nair, R. Krishna, and T. Menzies. "sampling" as a baseline optimizer for search-based software engineering. *IEEE Transactions on Software Engineering*, 45(6):597–614, 2018.

[2] M. Harman and B. F. Jones. Search-based software engineering. *Information and software Technology*, 43(14):833–839, 2001.

[3] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26:369–395, 2004.

dataset	characteristic	mean	median	mode	standard deviation
auto2	CityMPG+	22.37	21	18	5.62
	HighwayMPG+	29.09	28	26	5.33
	Weight-	3072.9	3040	3470	589.9
	Class-	19.51	17.7	15.9	9.69
auto93	Lbs-	2970.42	2803.5	2130	846.84
	Acc+	15.57	15.5	14.5	2.76
	Mpg+	23.84	20	20	8.34
china	N_effort-	4277.64	2098	296	7071
coc1000	LOC+	1013.05	1060.5	720	571.35
	AEXP-	2.97	3	2	1.2
	RISK-	6.68	5	0	6.37
	EFFORT-	30807.5	19642	33906	33883.81
coc10000	Loc+	1009.04	1012	100	574.75
	Risk-	6.59	5	0	6.04
	Effort-	30506.37	19697.5	4509	35435.43
health...0001-hard	MRE-	82.32	75.04	199	12.45
	ACC+	5.15	7.14	0	3.82
	PRED40+	22.1	25	25	13.52
health...0011-easy	MRE-	92.3	119.33	0	48.43
	ACC+	-8.53	-12.24	0	5.71
	PRED40+	17.79	0	0	34.13
nasa93dem	Kloc+	94.02	47.5	100	133.6
	Effort-	624.41	252	60	1135.93
	Defects-	3761.76	2007	2077	6145.06
	Months-	24.18	21.4	13.6	12.97
pom	Cost-	369.99	327.32	0	204.40
	Completion+	0.87	0.9	1	0.13
	Idle-	0.24	0.23	0	0.2
SSM	NUMBERITERATIONS-	30.94	7	5	94.53
SSN	PSNR-	44.53	45.91	45.98	6.47
	Energy-	1658	1258.09	0	1610.66

TABLE I
DATASET STATISTICS

	CityMPG+	Class-	HighwayMPG+	Weight-
sway1	26.02	13.75	32.38	2661.03
xpln1	27.07	16.17	33.14	2632.12
sway2	24.363	16.86	31.25	2845.97
xpln2	25.26	16.72	25	2781.65
top	37.16	9.26	41.75	2040.98

TABLE II
RESULTS FOR AUTO2.CSV

	Lbs-	Acc+	Mpg+
sway1	2240.94	16.80	29.51
xpln1	2416.26	15.44	26.40
sway2	2319.56	16.62	29.72
xpln2	2456.19	14.49	23.57
top	1998.07	19.77	40.76

TABLE III
RESULTS FOR AUTO93.CSV

	N_effort-
sway1	1800.38
xpln1	2619.90
sway2	1965.81
xpln2	2431.86
top	145.94

TABLE IV
RESULTS FOR CHINA.CSV

	LOC+	AEXP-	PLEX-	RISK-	Effort-
sway1	1027.74	3.05	2.88	5.08	29321.05
xpln1	913.11	2.67	2.73	5.7	27416.45
sway2	999.59	2.92	3.02	6.12	28323.93
xpln2	918.43	2.66	2.73	6.0	26910.59
top	1571.43	1.62	1.39	4.7	35116.16

TABLE V
RESULTS FOR COC1000.CSV

	Kloc+	Effort-	Defects-	Months-
sway1	70.52	428.92	2811.09	20.86
xpln1	72.85	450.05	2894.04	21.40
sway2	85.92	524.90	3289.46	22.04
xpln2	84.24	542.82	3374.83	22.59
top	4.59	18.29	143.51	8.24

TABLE VI
RESULTS FOR NASA93DEM.CSV

	MRE-	ACC+	PRED40+
sway1	74.71	7.40	19.215
xpln1	75.03	7.28	19.18
sway2	75.37	7.32	18.23
xpln2	75.03	7.27	19.24

TABLE VII
RESULTS FOR HEALTHCLOSEISSES12MTHS0001-HARD.CSV

	MRE-	ACC+	PRED40+
sway1	31.37	-0.39	57.53
xpln1	35.96	-0.46	53.85
sway2	26.03	-0.46	62.11
xpln2	35.96	-0.46	53.85

TABLE VIII
RESULTS FOR HEALTHCLOSEISSES12MTHS0011-EASY.CSV

	PSNR-	Energy-
sway1	44.48	1126.28
xpln1	44.29	1621.32
sway2	44.21	1612.90
xpln2	44.46	1640.16

TABLE IX
RESULTS FOR SSN.CSV

	NUMBERITERATIONS-
sway1	5.37
xpln1	10.20
sway2	6.27
xpln2	9.37

TABLE X
RESULTS FOR SSM.CSV

	LOC+	RISK-	EFFORT-
sway1	1004.94	5.22	26372.00
xpln1	604.81	4.02	17308.46
sway2	1006.16	4.57	24570.96
xpln2	454.17	2.71	13859.63

TABLE XI
RESULTS FOR COC10000.CSV

	Kloc+	Effort-	Defects-	Months-
10	70.52	329.06	2038.96	19.09
25	47.925	250.97	1743.88	17.87
50	64.64	311.22	2427	20.08
100	62.24	314.84	2325.34	19.14
200	94.33	602.88	3629.39	24.35
500	86.29	489.42	3323.69	22.12
1000	106.19	635.1	4117.55	25.47

TABLE XII
RESULTS FOR DIFFERENT SAMPLE SIZE FOR NASA93DEM.CSV

dataset	characteristic	all			sway1			sway2	
		all	sway1	sway2	sway2	xpln1	top	xpln2	top
auto2	CityMPG+	=	≠	≠	=	=	≠	=	≠
	HighwayMPG+	=	≠	≠	=	=	≠	=	≠
	Weight-	=	≠	≠	=	=	≠	=	≠
	Class-	=	≠	≠	≠	=	≠	=	≠
auto93	Lbs-	=	≠	≠	=	=	≠	≠	≠
	Acc+	=	≠	≠	=	≠	≠	≠	≠
	Mpg+	=	≠	≠	=	≠	≠	≠	≠
china	N_effort-	=	≠	≠	=	≠	≠	=	≠
coc1000	LOC+	=	=	=	=	=	≠	=	≠
	AEXP-	=	≠	=	=	=	≠	=	≠
	PLEX-	=	≠	=	=	=	≠	=	≠
	RISK-	=	≠	=	=	=	=	=	=
	EFFORT-	=	≠	≠	=	=	≠	=	≠
coc10000	Loc+	=	=	=	=	≠	n/a	≠	n/a
	Risk-	=	≠	≠	=	=	n/a	≠	n/a
	Effort-	=	≠	≠	=	≠	n/a	≠	n/a
health...0001-hard	MRE-	=	≠	≠	=	=	n/a	≠	n/a
	ACC+	=	≠	≠	=	≠	n/a	=	n/a
	PRED40+	=	≠	≠	=	=	n/a	=	n/a
health...0011-easy	MRE-	=	≠	≠	=	=	n/a	≠	n/a
	ACC+	=	≠	≠	=	≠	n/a	=	n/a
	PRED40+	=	≠	≠	=	=	n/a	≠	n/a
nasa93dem	Kloc+	=	=	=	=	=	≠	=	≠
	Effort-	=	≠	=	=	=	≠	=	≠
	Defects-	=	≠	=	=	=	≠	=	≠
	Months-	=	=	=	=	=	≠	=	≠
pom	Cost-	=	≠	≠	=	=	n/a	≠	n/a
	Completion+	=	=	=	=	≠	n/a	≠	n/a
	Idle-	=	≠	=	=	≠	n/a	≠	n/a
SSM	NUMBERITERATIONS-	=	≠	≠	=	≠	n/a	≠	n/a
SSN	PSNR-	=	=	=	=	=	n/a	=	n/a
	Energy-	=	≠	=	=	≠	n/a	=	n/a

TABLE XIII
RESULTS FOR EFFECT SIZE TEST AND SIGNIFICANCE TEST