Algorithms for Software Quality Optimization

Jamie Wood
Lane Department of CS&EE
West Virginia University, USA
jwood13@mix.wvu.edu

Andrei Perhinschi
Lane Department of CS&EE
West Virginia University, USA
aperhins@mix.wvu.edu

ABSTRACT

Software quality optimization is an area of research that has received much attention, mostly because there exists a large number of algorithms, hence many choices, which can be applied to this particular problem. Quality optimizers are programs that find relationships between project metrics and quality metrics, such as effort and software defects, with the goal of identifying which project attributes can be changed, and by how much, to elicit an improvement in the quality metric. This paper is based on previous research on an experimental quality optimizer called $\mathcal{W}2$ [1]. $\mathcal{W}2$ implements contrast set learning, that is an algorithm which looks at 2 populations and determines what separates them.

Though $\mathcal{W}2$ implements an efficient, fast, linear time algorithm, there are concerns about the stability and consistency of the recommendations it makes. In order to establish the degree to which $\mathcal{W}2$'s results are stable we attempt to replicate the original findings. Furthermore, we explore how the amount and type of historical data affects the consistency of $\mathcal{W}2$'s results. To this end we investigate $\mathcal{W}2$'s performance on several effort estimation datasets that were used in previous research, as well as a number of defect prediction datasets that had not been considered before. We conclude with a short list of areas where the system can be improved upon, along with recommendations on what may be possible as well as some specific topics of interest we believe future research should address.

Keywords

Effort Estimation, Analogy, Optimization, Software Quality, Case Based Reasoning

1. INTRODUCTION

Learning from our mistakes and successes is an often-ignored human ability. We remember past experiences and events, analyze the conditions present before and after, and finally we attempt to extract some sort of understanding as to the effect of initial conditions on the end result. We do this because we crave control. Being able to predict the effect of our actions allows us to change and adapt what we do in such a way as to bring about a desired, usually more efficient, more optimized, end result. Software quality optimizers attempt to do just this by employing a number of different methods.

From the inception of computer software there have existed a myriad of software projects, each with their given deadlines, budgets, number of workers, etc. Each of these past projects has its own history, more specifically, a number of factors, or metrics, that affected the final outcome of the project in some way. Such historical information can be collected and processed in order to extract usable data regarding the relationships between project metrics and project outcome. Contrast set learning does just that by looking at the best performing projects and comparing them to the rest. The attributes that helped a majority of the best become the best are then adopted in current, similar, projects.

The question then becomes how to best put this information to use regarding new projects. There are several different ways of doing this, such as parametric models, however $\mathcal{W}2$ uses a method called case based reasoning. CBR uses the historical information itself to extract meaningful data, bypassing the need for a parametric model [1].

From previous research conducted on $\mathcal{W}2$'s performance and results [2] we begin our work aware of the following:

- $\mathcal{W}2$ is a clear performance improvement over its predecessor, W.
- $\mathcal{W}2$ matches or outperforms other quality optimization methods.
- $\mathcal{W}2$ attains fast run times on the datasets it's been tested on.
- $\mathcal{W}2$ presents a lightweight system footprint.
- $\mathcal{W}2$'s lack of parametric modeling lends it high flexibility in terms of different datasets that can be accepted.

These previous findings act as the background against which we conduct some of the same tests, as well as several new ones, to determine $\mathcal{W}2$'s level of performance and usability.

2. OVERVIEW

2.1 Contrast Set Learning

Contrast set learning is a quality optimization technique that provides the ability to distinguish the differences between two populations. The desire to incorporate CSL decision making into case based reasoning quality optimization software was the motivation for $\mathcal{W}$'s initial development. In essence, $\mathcal{W}$ was designed as a case based reasoner with the added ability of performing contrast set learning. $\mathcal{W}$'s case relevancy filtering was improved upon in $\mathcal{W}2$ by replacing the k-th nearest neighbor computations with a simple, linear time, overlap method.

2.2 $\mathcal{W}2$ Algorithm

To perform its function, $\mathcal{W}2$ requires two sets of data: a training dataset and a project description or test case. Once this information becomes available, $\mathcal{W}2$ randomly splits the dataset into 67% training cases and 33% testing cases. The training instance set is then put through relevancy filtering so as to retrieve only those cases with enough overlap to the test case. The resulting set of projects is sorted using some metric of utility and split into a best set and a rest set. $\mathcal{W}2$ then proceeds to generate a contrast set that it uses to find the specific attributes which most push the test project towards

NASA93 x30				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	flight	22%	21%	
months	ground	11%	10%	
months	osp	40%	51%	
months	osp2	33%	50%	
defects	flight	40%	54%	
defects	ground	34%	40%	
defects	osp	71%	74%	
defects	osp2	64%	75%	
effort	flight	51%	60%	
effort	ground	42%	44%	
effort	osp	69%	74%	
effort	osp2	64%	71%	

Figure 1: Comparing defect, effort, and month estimation reduction percentages over 30 runs of $\mathcal{W}2$

the desired utility measures. At this point $\mathcal{W}2$ takes the attribute recommendations from its previous step and estimates their success at improving the test instance. Upon exiting, $\mathcal{W}2$ outputs the attribute recommendations it thinks are most likely to lead the test project towards optimization.

3. SYSTEM STATE

To demonstrate the current state of the $\mathcal{W}2$ system, we ran a series of trials comprised of tests similar to the ones performed previously, as well as several tests designed to reveal more information about the way $\mathcal{W}2$ performs across a multitude of situations and conditions.

3.1 Basic Testing

In order to establish a frame of reference against which to compare any subsequent results, we began by running $\mathcal{W}2$ through a series of tests similar to what had been tried before. The results in figure 1 can be said to be similar to what has been obtained previously. More specifically, for osp and osp2, the results were numerically rather close. Additionally, we explored the same 4 dataset/project combinations collecting information from 60 trials as opposed to just 30.

4. CONCERNS

While working with $\mathcal{W}2$, one serious issue that would hinder the usability of this software outside of a laboratory setting, becomes immediately apparent. The existing $\mathcal{W}2$ codebase is plagued

NASA93 x60				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	flight	11%	9%	
months	ground	13%	12%	
months	osp	33%	41%	
months	osp2	32%	48%	
defects	flight	24%	29%	
defects	ground	34%	43%	
defects	osp	63%	74%	
defects	osp2	62%	68%	
effort	flight	35%	41%	
effort	ground	37%	39%	
effort	osp	62%	70%	
effort	osp2	67%	67%	

Figure 2: Comparing defect, effort, and month estimation reduction percentages over 60 runs of $\mathcal{W}2$

NASA93 Flight 3x Stability				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	runs 1	33%	-5%	
months	runs 2	15%	41%	
months	runs 3	16%	12%	
defects	runs 1	66%	3%	
defects	runs 2	22%	48%	
defects	runs 3	0%	55%	
effort	runs 1	68%	34%	
effort	runs 2	60%	51%	
effort	runs 3	46%	56%	

Figure 3: Comparing defect, effort, and month estimation reduction percentages over 3 runs of $\mathcal{W}2$

by inconsistencies in the recommendations it suggests. The instability of $\mathcal{W}2$'s results shows up both when performing individual runs as well as when compiling statistical results from multiple runs. Consistency issues persist across different datasets as well. We believe the combination of inconsistency across individual runs and inconsistency present across statistical results from multiple trials can make it difficult to correctly interpret results.

4.1 Single and Multiple Trial Inconsistancy

When performing single runs, $\mathcal{W}2$ comes up with, sometimes wildly, varying results from run to run. A trial with median and spread reductions of well over 50% will be followed by a trial with negative reductions. Whenever $\mathcal{W}2$'s recommendations lead to a negative reduction in utility metrics, that is an increase in utility metrics, we consider these recommendations faulty. Such trials are uncommon, however they do occur. There also does not appear to be any discernible pattern as to when $\mathcal{W}2$ will return a faulty recommendation. The same thing happens after running $\mathcal{W}2$ a number of times, in our case 30, and noting the medians of the median reduction and spread reduction sets collected from each run, the same instability becomes noticed upon repeating the same number of runs. Figure 3 demonstrates this by listing the results from 3 runs of 1, 30, and 60 trials. The values for the runs of 30 and 60 trials are the medians of the values collected from the trials.

NASA93 Flight 15x Stability				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	runs 1	21%	31%	
months	runs 2	20%	25%	
months	runs 3	3%	20%	
defects	runs 1	37%	55%	
defects	runs 2	39%	34%	
defects	runs 3	0%	14%	
effort	runs 1	51%	53%	
effort	runs 2	47%	48%	
effort	runs 3	36%	37%	

Figure 4: Comparing defect, effort, and month estimation reduction percentages over 15 runs of $\mathcal{W}2$

NASA93 Flight 75x Stability				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	runs 1	12%	8%	
months	runs 2	15%	23%	
months	runs 3	17%	26%	
defects	runs 1	27%	33%	
defects	runs 2	33%	43%	
defects	runs 3	40%	47%	
effort	runs 1	49%	38%	
effort	runs 2	48%	52%	
effort	runs 3	58%	61%	

Figure 5: Comparing defect, effort, and month estimation reduction percentages over 75 runs of $\mathcal{W}2$

4.2 Increasing Number of Trials

While there is nothing one can do about instability from one trial to another without changing the software, over many runs statistical analysis can bring results to the surface. At first we assumed it would be possible to remove the consistency issues by running $\mathcal{W}2$ a large enough number of times and looking at the medians of the me-

dian and spread reduction sets collected from each run. To find out if this would work we increased the number of trials incrementally from 30 to 100 runs without noticing any improvement in result consistency. While it may be possible that all one needs to do to avoid this lack of consistency is run $\mathcal{W}2$ a large enough number of times, we could not find it in our tests. If there is a statistical limit the results tend towards, more than 100 trials would be required to find it. We did learn that the number of runs resulting in faulty recommendations, that is recommendations that lead to increased utility metrics, increases as more and more trials are performed.

4.3 Inconsistency Across Datasets

The issue of inconsistency remains present while using $\mathcal{W}2$, regardless of dataset choice. We cannot state that the dataset used has no influence on consistency, only that inconsistency was observed among all datasets tested. Future research applying different types of historical data will show if and how $\mathcal{W}2$'s performance is affected.

5. FUTURE WORK

0;136;0cAfter having experimented with $\mathcal{W}2$ we were impressed with its execution speed and generally pleased with the reductions stemming from its recommendations. That being said, $\mathcal{W}2$ is not without its problems. There are some noticeable issues we ran into that we feel require being explored in more detail.

5.1 Result Consistency

As discussed briefly above, we consider the reliability of $\mathcal{W}2$'s results its only real shortcoming at this point. To apply a quality optimizer's recommendations and feel confident of this decision, we have to be certain that the software is capable of consistently providing us with the best possible results for the project at hand. In order to attain a certain level of consistency within our results we believe there might be several possible avenues worth exploring further.

5.2 Larger Datasets

One possible way of increasing the reliability of our results is using much larger datasets. The reasoning here being that as we increase the amount of historical instances available, we increase the number of projects that highly overlap our test case. It follows that if the project metrics present in the learning data are all closer to our test instance, the best set should be better and the results we obtain in the end should be more precise. The preliminary tests we ran with large datasets were used only to record runtimes. We took the largest effort estimation dataset available, namely "china", and copied and pasted its data back into itself until we ended up with a series of files containing between 2 and 512 times the amount of historical instances in the original dataset. Simply replicating the data within a given dataset is assumed to have some effect on the results, however since we are testing for run times we are not concerned with results at this point. Figure 4 graphically demonstrates the linear behavior. Testing whether or not there is an improvement in results with larger datasets remains to be explored.

5.3 Size of Best and Rest Sets

The only method we found that allowed us to affect the consistency of the recommendations was to modify $\mathcal{W}2$'s K1 and K2 parameters. K1 represents the number of instances in the best set, while K2 represents the number of instances in the rest set, therefore the sum of these two values is the number of instances that most overlap the test project. All of the previous results obtained from $\mathcal{W}2$ used a value of 20 for K1+K2, where K1=5, K2=15 [2]. What we found is that increasing the size of the best and rest sets has a very positive effect on the stability of $\mathcal{W}2$'s results. Not only did both the median and spread values became consistent among separate instances of 30 runs, but the reduction values themselves increased dramatically. During our trials we used the values 10 and 30 for K1 and K2 respectively. Figure 5 contains the results from running nasa93/ground with 100 trials 8 times. Half of the trials were conducted with K1=5, K2=15, these are the first 3 columns. The last 3 columns represent the other half of the trials that were conducted with K1=10, K2=30. Figure 5 clearly shows that not only did the reductions increase in every individual utility metric, but consistency from one set of 100 runs

to another improved as well. A curious observation is that the new K values prevent the minimum median and spread reductions from falling as dramatically as they would otherwise. While no objective run time measurements were taken, no performance decrease was noticed with this minor change in place. Despite this, we recommend run time testing with modified K1, K2 values be a part of any future work focusing around $\mathcal{W}2$.

5.4 Further Considerations on Best/Rest

Given the observations outlined in section 5.3, an interesting question arises with respect to changing the sizes of the best and rest sets. We believe it is worth investigating whether or not there exists a pair of K1 and K2 that maximizes the consistency and the magnitude of $\mathcal{W}2$'s reduction values. Furthermore, if it turned out that such a pair of values exists, the relationship between dataset and these values should be explored. Such a relationship, if known, would allow us to pre-process the data and dynamically set K1 and K2 at runtime such that consistency and metric reduction are both maximized per dataset.

There is also the possibility that the improvements we are seeing in our recommendations are in fact the result of deeper issues within the system and not indicative of an overall positive effect. Further experimentation is required to determine which is the case.

5.5 Effects of Training Data Quality

One important consideration we have to keep in mind is that any results obtained can only be as trustworthy and reliable as the historical data used in training. Having said that, there are several areas in which the data we use for learning can be lacking in. Examples of what we consider to be poor quality data include very short datasets as well as datasets containing utility metrics with very limited ranges of values.

5.6 Very Short Datasets

We observed the effects of both types of data described above during our exploration of defect prediction datasets. Very short datasets, containing a handful of historical instances, are not only not well suited to our goal of extracting information,

NASA93 K1=5 k2=15				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	flight 1	22%	21%	
months	flight 2	12%	19%	
months	flight 3	6%	12%	
defects	flight 1	40%	54%	
defects	flight 2	29%	42%	
defects	flight 3	17%	29%	
effort	flight 1	51%	60%	
effort	flight 2	42%	47%	
effort	flight 3	40%	60%	

Figure 6: Comparing defect, effort, and month estimation reduction percentages over 30 runs of $\mathcal{W}2$ with default K values

NASA93 K1=10 k2=30				
Goal	Change	Median Reduc	Spread Reduc	Reduction Quartiles 50%
months	flight 1	38%	59%	
months	flight 2	38%	59%	
months	flight 3	37%	54%	
defects	flight 1	68%	81%	
defects	flight 2	68%	81%	
defects	flight 3	69%	79%	
effort	flight 1	75%	82%	
effort	flight 2	75%	82%	
effort	flight 3	68%	74%	

Figure 7: Comparing defect, effort, and month estimation reduction percentages over 30 runs of $\mathcal{W}2$ with double k values

but also result in $\mathcal{W}2$ terminating with a "divide by 0" error a majority of the time. Switching to a dataset with regular length, we observe the "divide by 0" errors disappear. To perform our tests on the effects of short datasets, we took a working, effort estimation dataset and removed all but the first 4 historical instances. We alternated between using the original data and the shortened copy all while keeping the test project file constant. The main conclusion we can draw from using very short datasets is that $\mathcal{W}2$ could benefit greatly from the addition of some rudimentary error prevention. Ideally, $\mathcal{W}2$ would never accept running with a dataset shorter than an established minimum length.

5.7 Limited Utility Metric Ranges

Another aspect of data quality became clear to us as we tried using different defect prediction datasets. What we found is that for a dataset containing a utility metric with a range of only a handful of values, the median and spread reductions will almost

always be 0. We first noticed this situation with the "Ant" dataset. As we used the single utility metric "bug", we consistently obtained no reductions in median or spread. Attempting to understand why our results were so reliably bad, we took a look at the raw data and noticed our utility metric had a range from 0 to 3 containing a mere 4 values. To ascertain whether or not this was the cause of our results, we applied the same technique described in section 5.6. We took a working, effort estimation dataset, and modified its contents to the point at which the range of its utility metric contained only 5 values. We then ran $\mathcal{W}2$ against the original and the modified datasets and observed that over any number of runs the reduction percentages consistently stay at 0.

The limited range exhibited by the defect prediction data's utility metric is either an indication of poor quality or the fact that the data is unsuited to reducing the metric we chose. In this case we considered the data to be of poor quality. The defect datasets contain what appear to be historical instances of projects with very low numbers of bugs. Whereas anywhere else bug-free software is a good thing, the absence of any instances with significant numbers of bugs makes it impossible for $\mathcal{W}2$ to gauge which attributes could lead to improvements. It follows that for the purpose of minimizing the number of bugs in software, several of the defect prediction datasets we tried contain poor quality data. If the utility metric is already reduced as far as it will go, there is nothing left to optimize.

This is another problem area that could be taken care of by a level of error protection built into the system. $\mathcal{W}2$ could pre-process the historical data and make sure the utility metrics have a wide enough range of values so as to avoid running into this issue.

5.8 Effects of Training Data Type

What do we mean when we use the term "type" while discussing datasets? In this context we apply the term "type" to those characteristics which we observe due to what the data in question represents. One clear example of such a characteristic is the number of attributes per instance contained in a given dataset. Such wider datasets could impact the performance of the system in unexpected ways. There may also be cases in which the ranges exhibited by the values of whatever utility metrics are used depend on what the data represents and not its quality, as was discussed in section 5.7.

Considering the above points, we believe that, besides its quality, it may be possible for the type of data used to influence how well the system performs and that this should be a topic for research in future work involving $\mathcal{W}2$.

6. CONCLUSION

Taking into account the inherent inconsistency apparent in the results obtained from $\mathcal{W}2$, we believe to have so far demonstrated results similar to the original findings. We therefore conclude that $\mathcal{W}2$ is a promising system currently being hampered by several issues:

- Inconsistency in results.
- Lack of error protection.
- Lack of understanding as to how $\mathcal{W}2$ scales to larger amounts of historical data.
- Lack of understanding as to how $\mathcal{W}2$ performs with other types of data.
- Somewhat confusing codebase makes for difficult maintenance.

With the above issues in mind, we identify several areas that warrant further exploration as part of ongoing $\mathcal{W}2$ research:

- Best/Rest set sizes and their effects on results.
- Optimum best/rest set sizes.
- Effect dataset type has on results.
- Larger datasets.

7. REFERENCES

[1] J. Kolodner. An Introduction to Case-Based Reasoning. Artificial Intellegence Review, 6, 3–34, 1992.

[2] A. Brady, J. Keung, and T. Menzies. Algorithms for Software Quality Optimization. In Promise '10, 2010.