Empirical Software Metrics for Benchmarking of Verification Tools

1 Introduction

This work chose a set of software metrics that they used to train an SVM to decide which tool from a set of verification tools should be used to verify a program.

2 Source Code metrics for software verification

These metrics fall into three categories: variable role, loop patterns, and control flow. Variable roles include array index, counter, iterator, or pointer to a struct. Loop patterns mean bounded loops (loops that will run a bounded number of times), loops that can be determined to terminate (find possible values of vars in the loop, find what conditions lead to ending the loop, determine if the set of var values will guarantee to termination), a more relaxed version of the previous loop (using heuristics they can determine these), and all other loops (no real idea if it will terminate). Control flow metics mean how many basic blocks in a CFG, the highest indegree of any node in a CFG. ratio of call expression with function pointers as arguments, the ratio of parameters to such call expressions have function pointer as arguments, and direct recursion calls.

To show these metrics have merit, they look to the developers reports of the tools to show that the developers themselves say certain constructs are strengths or weaknesses of their techniques. In previous works, they've shown that variable roles and loop patterns are good predictors for verification success.

3 A portfolio solver for software verification

This section begins by describing the basics of support vector machines and supervised learning in general. They then describe the SV-Comp and it's rules.

They construct three portfolios. TP^{time} chooses the tool that has the lowest predicted runtime. TP^{mem} chooses the tool with the lowest predicted memory cost. TP^{prob} chooses the tool with the highest probability to return the expected answer. Understandably, TP^{prob} performs the best at the SV-Comp, and this is the portfolio they choose to proceed with. They also have some talk about how they deal with data imbalances through weighting

4 Experimental Results

They begin by talking about how SV-Comps scoring has changed and how different tools would have won or lost based on these scorings. Specifically, incorrect answers have become more costly. They introduce a chart which shows the ratio of correct to incorrect responses.

They then proceed to look at how their tool performs in these competitions. They find that they would win SV-Comp 2014, 2015, and 2016. They find their portfolio doesn't simply rely on a few tools and actually looks at many. They discuss why the cases where their tools may fail. Specifically, they look at cases where small changes in programs may cause tools to incorrectly answer. They find that their technique has a small cost in running.

5 Related Work

They discuss other uses for algorithm selection, such as SMT solvers and how their approach is novel.