Sufficient Mutation Operators for Measuring Test Effectiveness

Akbar Siami Namin and James H. Andrews
Department of Computer Science
University of Western Ontario
London, Ontario, Canada
{asiamina, andrews}@csd.uwo.ca

Duncan J. Murdoch
Department of Statistics and Actuarial Sciences
University of Western Ontario
London, Ontario, Canada
murdoch@stats.uwo.ca

30th International Conference on Software Engineering, ICSE 2008, Leipzig, Germany
May 2008

Outline

- Motivation
- A Quick Review of Mutation
- Sufficient Set Problem
- Applicable Statistical Techniques
- Experimental Procedure
- Data Analysis
- Conclusion
- Research Directions

Motivation

- >50% of the development budget is still allocated for testing
- Software systems are becoming more complex

Need More Effective and Less Expensive Testing Techniques

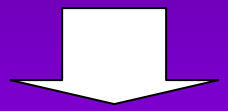
- Measuring the Cost: Easy
  - E.g., # of test cases needed
- Assessing the Effectiveness!!!

How?

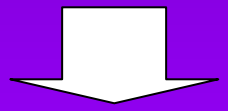
Motivation

Assessing the Effectiveness of Testing Techniques

- Empirical Investigation and Experimentation
  - Testing technique A is more effective than testing technique B if A detects more faults than B for a set of subject programs with known faults



Need a set of subject programs accompanied by some faults



Using Mutation to Generate Faulty Versions

Mutation

Terminologies

- Mutation – Modifying the source code slightly
- Mutation Operator – Mathematically well-defined functions that transforms an element of the source code to another element in the same class
- Mutant – The source code resulting from applying a mutation operator
- Killed Mutants – There is at least one test case in the test suite that detects the mutant
  - Running the test case on both the gold and the mutant produces different output
- Alive Mutants – No test case in the test suite is able to detect the mutant. But by adding the required test cases it might be detected
- Equivalent Mutant – There is absolutely no test case to detect the mutant

Mutation

Faults vs. Mutants

- Andrews et al. [2005, 2006] showed that:
  - ➢Mutants can be a good representation of real faults
  - ➢ They have can have high correlations for both random and coverage-based test suites

Mutant ↔ Fault

Mutation

An Expensive Procedure

- > 100 mutation operators
  - E.g., 137 lines of code => 4935 mutants generated by 108 mutation operators
- Makes mutation infeasible

Sufficient Set Problem [Offutt et al., 1996]

"Search for a subset of mutation operators that can be used instead of all mutation operators in measuring the adequacy of test cases"

Sufficient Set Problem

Related Work

- Mutation Testing
  - Used to enhance a test suite [Hamlet 1977; De Millo et al. 1978]
  - Mutation Tools
    - Proteum, MuJava
- Sufficient Operators
  - Compared 22 operators with 2 operators [Wong 1993]
  - Compared 22 operators with 4 subsets [Offutt et al. 1996]
  - Defined guidelines for selecting [Barbosa et al. 2001]
- Generating Faulty Versions
  - Hand-seeding [Frankl and Weiss, 1993]
  - Reintroducing [Frankl and Iakounenko, 1998]
  - Mutation [Daran et al., 1996]

Sufficient Set Problem

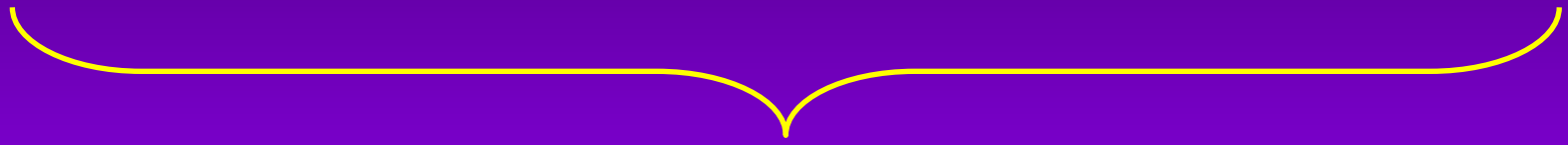
Why is Our Work Different?

1. Using larger subject programs with complex data structures
   - 137 – 513 lines of code
2. Sufficient set for a wide range [0, 1] of prediction for mutation score
3. A variable reduction problem
4. Identifying important operators through linear regressions

Sufficient Set Problem

Our Definition

- We refine sufficient set problem as:
  - Finding a subset of mutants and a multiple linear regression model that can predict the behavior of all mutants



The problem is transformed to a statistics problem

Sufficient Set Problem

The Scheme

For each test suite s, we compute:

1. Mutants Detection Rate (AM) (Generated by all operators):

$$AM = \frac{\#\text{Killed Non - Equivalent Mutants}}{\#\text{All Non - Equivalent Mutants}}$$

2. Mutant Detection Rate (Am_i) (Generated by operator i):

$$Am_i = \frac{\#\text{Killed Non - Equivalent Mutants of Operator i}}{\#\text{All Non - Equivalent Mutants of Operator i}}$$

Sufficient Set Problem

Rephrasing The Problem

The Problem is to find a linear model:

$$AM = k + c_1 Am_1 + c_2 Am_2 + \ldots + c_n Am_n + \varepsilon$$

- The right hand side involves operators with minimum costs
- The prediction should be accurate

Sufficient Set Problem

Cost of Operators

- Sufficient set problem
  - A special case
    - Unequal costs are associated with operators
  - Cost of operator i

$$\text{cost}(i) = \frac{\#\,\text{Non - Equivalent Mutants Generated by i}}{\#\,\text{All Non - Equivalent Mutants}}$$

Possible Statistical Techniques

- What Statistics offers:
  - Variable Reduction Techniques
    - Correlation Analysis
      - Based on the correlation between operators
    - Cluster Analysis
      - Based on the distance of two operators
  - All-Subset Selection and Multiple Linear Models

All Subset Selection

- The Idea of Prediction and Applying Multiple Linear Regression
- Output: A set of good models (active sets: important variables)
- Variants
  - Exhaustive
  - Forward (Backward) Stage-wise
  - Lasso
  - Least Angle Regression
    - Intermediate steps
    - Better performance and answer set
    - Variables with equal costs

Cost-based Least Angle Regression
CbLARS

- Least Angle Regression (LARS)
  - Variables have equal costs
- Cost-based Least Angle Regression (CbLARS)
  - Developed as a collaboration between two departments (Statistics and Computer Science) at the University of Western Ontario
  - A modification of Original Least Angle Regression
  - Variables are associated with unequal costs

Experimental Procedure

The Artifacts

- Subject Programs: the Siemens set
  - Consists of seven programs
    - Net Lines of code: 137 => 513
    - #test cases: 1052 => 5542
- Mutant Generator: Proteum (Maldonado et al.)
  - A Mutant generator for C
  - Implements 108 mutation operators

Experimental Procedure

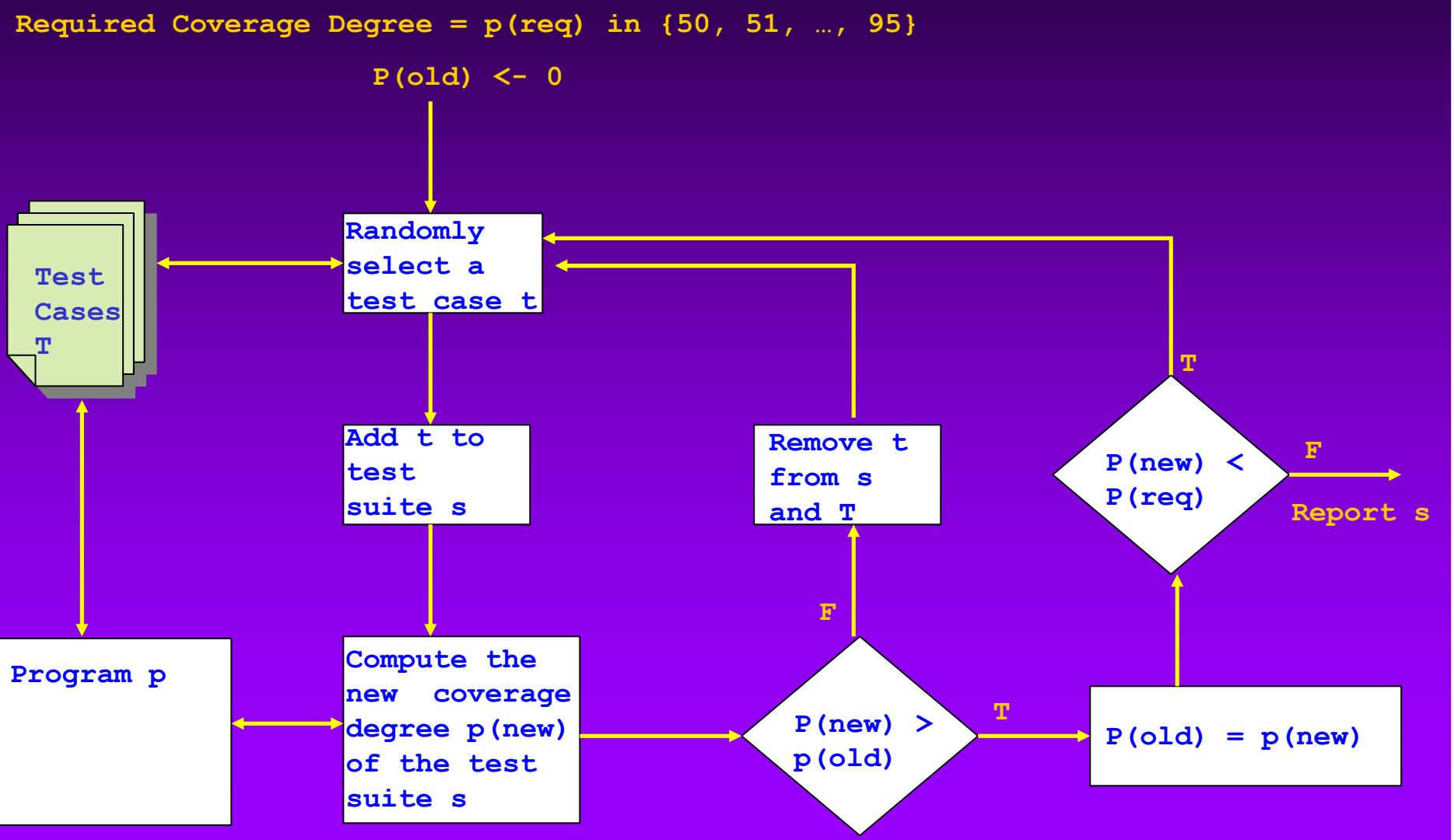
A Summary

Subjects	Lines of Code	#Mutants	#Selected	#Equivalent
tcas.c	137	4935	4935	0
totinfo.c	281	8767	1958	218
schedule.c	296	4130	1964	204
schedule2.c	263	6552	1964	467
printtokens.c	343	11741	1966	415
printtokens2.c	355	10266	1963	21
replace.c	513	23847	1969	0
		70238	16719	1325

Total #Non-Equivalent Mutants = 15394

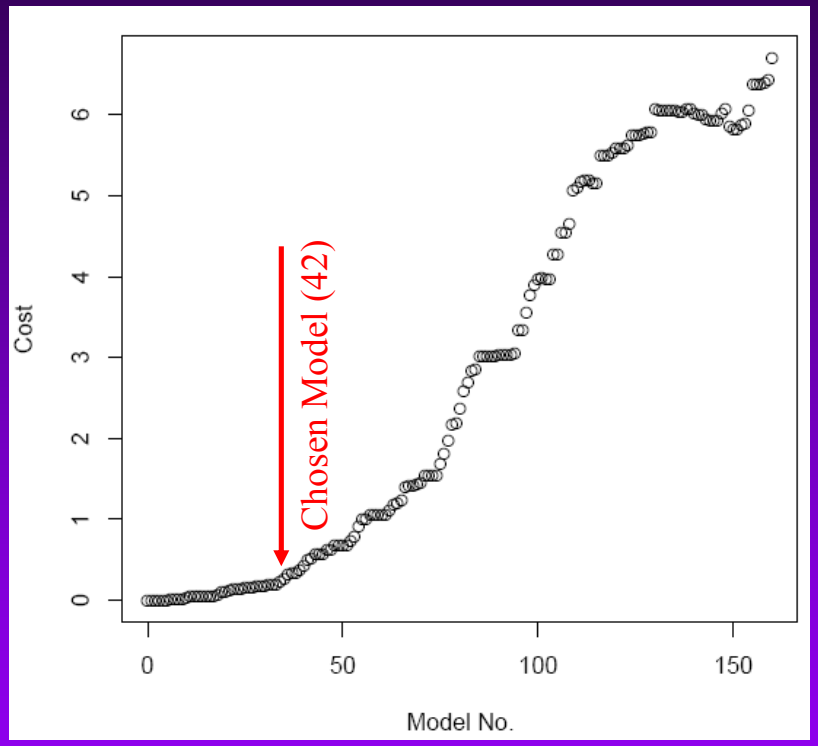
Experimental Procedure

Test Suite Generation



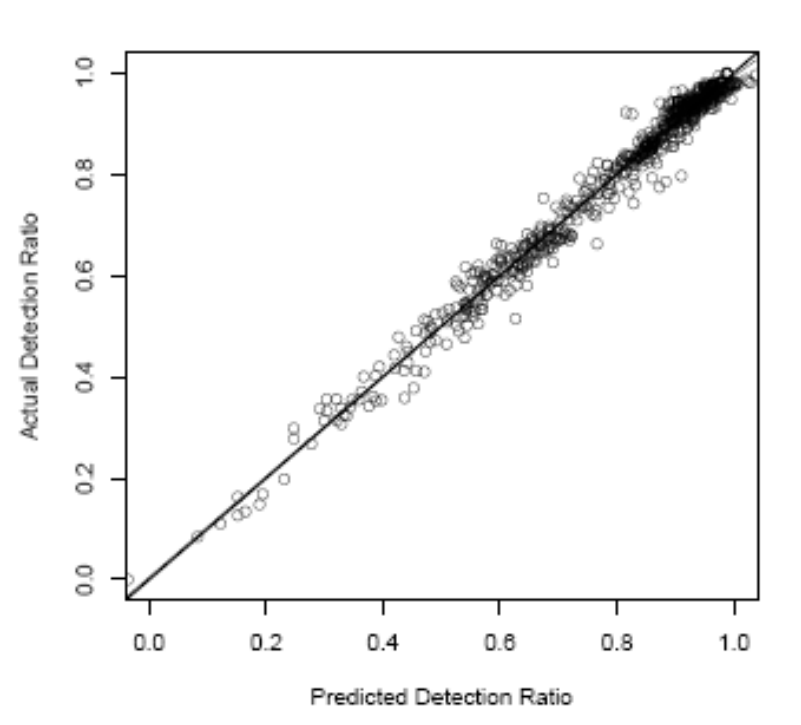
Data Analysis

CbLARS – The First Model with Adjusted $R^2 \geq 0.98$



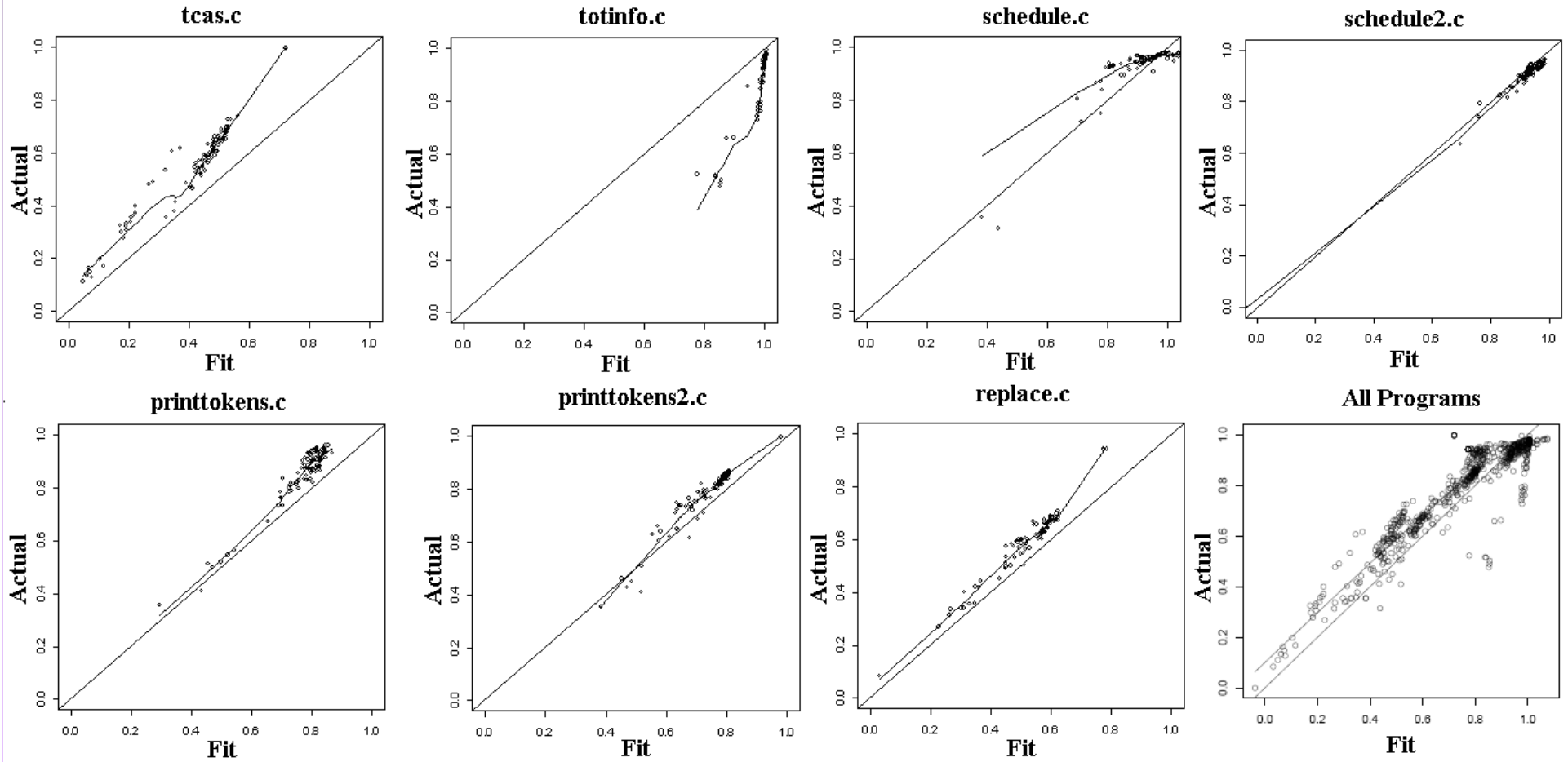
Data Analysis

CbLARS – Fitted vs. Actual



Data Analysis

CbLARS - CrossValidation



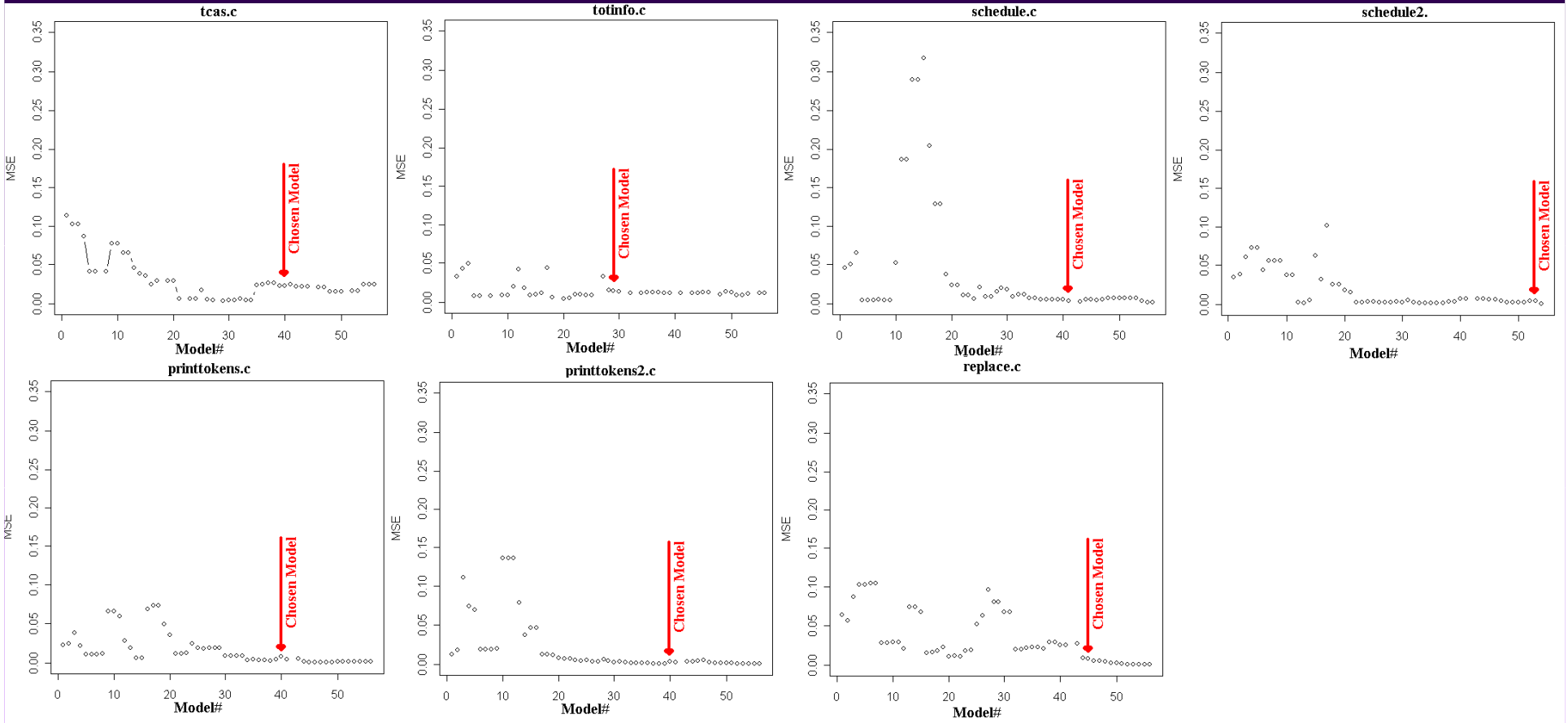
Data Analysis

CbLARS – Seven Fold Cross Validation

Subjects	MSE	r2	Spearman	Kendall
printtokens.c	0.008	0.930	0.769	0.601
printtokens2.c	0.003	0.924	0.967	0.849
replace.c	0.008	0.963	0.973	0.876
schedule.c	0.003	0.747	0.873	0.725
schedule2.c	<0.001	0.982	0.911	0.755
tcas.c	0.022	0.936	0.965	0.866
totinfo.c	0.015	0.841	0.900	0.765

Data Analysis

CbLARS – Overfitting Check with MSE

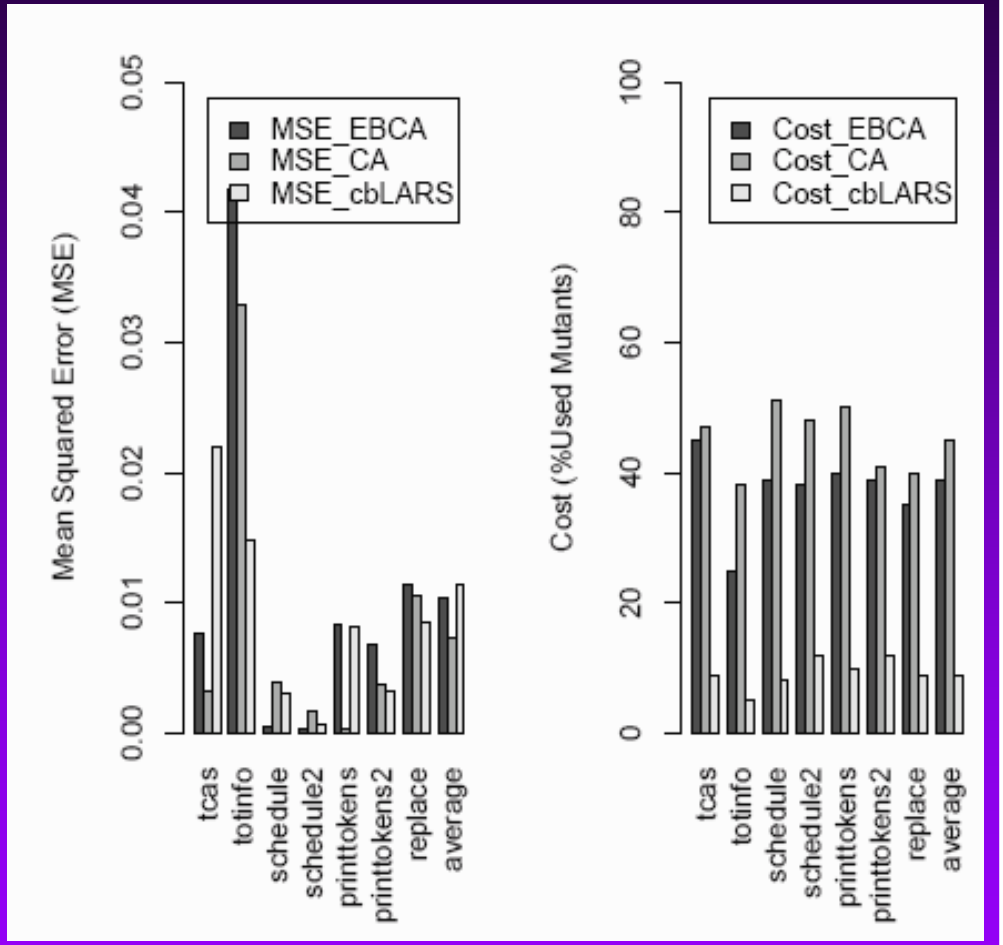


Conclusion

Comparison

Technique	# Operators	# Mutants	% Mutants
Correlation	52	6592	42.82%
Cluster	58	8596	55.83%
CbLARS	28	1139	7.39%

- CbLARS outperforms others
- The cut-off threshold value for correlation (k = 0.9) might be too high for this problem
- Correlation $\subset$ Cluster
- 22 operators in common



Conclusion

Interesting Observations

- No Mutation Operators for Mutating Constants Selected!!
  - E.g., Cccr Constant for Constant Replacement
  - Cccr generates many similar mutants
- 50% of negation operators were selected
  - E.g., ArgLogNeg Insert Logical Negation on Argument

Research Directions

- Identifying Sufficient Set for Coverage-based Test Suites
- Sufficient Set for Object-oriented Languages

Thank You