NC STATE UNIVERSITY

Replicating Findings: Fault-Proneness in Coverage-Based Test Case Prioritization

Final Presentation 11/25

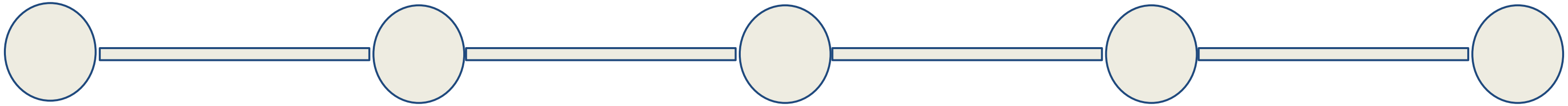
Sanjit Verma
Arul Sharma
Keith Tran
Alex Taylor

Project Description

Objective: Verify the claims and results of ***Incorporating fault-proneness estimations into coverage-based test case prioritization methods*** *(Mahdieh et al. 2020)*

- **Scope:** Evaluate and compare the performance of the total and additional strategies for test case prioritization methods, including their modified versions, as discussed in the paper.
- **Evaluation Metric:** Use Average Percentage of Faults Detected (APFD) as the primary performance metric for comparison.
- **Outcome:** Analyze and report differences in APFD values to evaluate the effectiveness of incorporating fault-proneness estimations.

Weekly Deliverables



Week 1

Defined Project Scope

Complete Literature review to identify algorithms

Document each algorithm’s strengths, limitations, and evaluation criteria

Week 2

Redefined Project Scope to evaluate algorithms instead of reimplementing

Selected 2 of the 9 papers to evaluate (MS 1)

Designed a common framework to apply each algorithm consistently (MS 1)

Week 3

Setting up the pipeline to run the algorithms, fixing any bugs with the outdated libraries

Generating APFD results for each algorithm (MS 2)

Week 4

Reproduce results on small libraries covered in the paper (MS 3)

Week 5

Analyze APFD result graphs and compare results for larger libraries (MS 4)

Validate our findings and compare them to the research paper (MS 5)

Cleaning up the Github repo and creating the final presentation

Paper Motivation

- Traditional TCP methods rely solely on code coverage, which can be suboptimal
- Incorporating fault-proneness estimations improves prioritization by focusing on parts of the code most likely to contain faults
- TCP is a topic covered in the class, we thought it would be interesting to dive deeper into it
- Paper evaluates methods on real world projects, improving chances of reproducibility
- Personal interests in trying to reproduce rather newer findings (2022)

Algorithms

Modified Total Strategy:

- Test cases are sorted due to their **total coverage** so that the first test case has the highest total coverage
- Uses all units, covered and uncovered, and is not iterative
- **Simple** but efficient

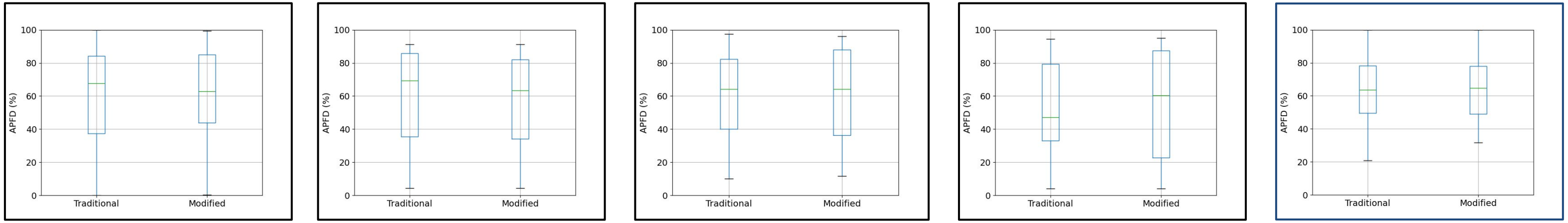
Modified Additional Strategy:

- Orders test cases using a **greedy** algorithm, selecting the test case with the **highest fault-based coverage** over the remaining **uncovered code areas** as the next test case.
- Uses only uncovered code areas, is iterative
- Higher time complexity

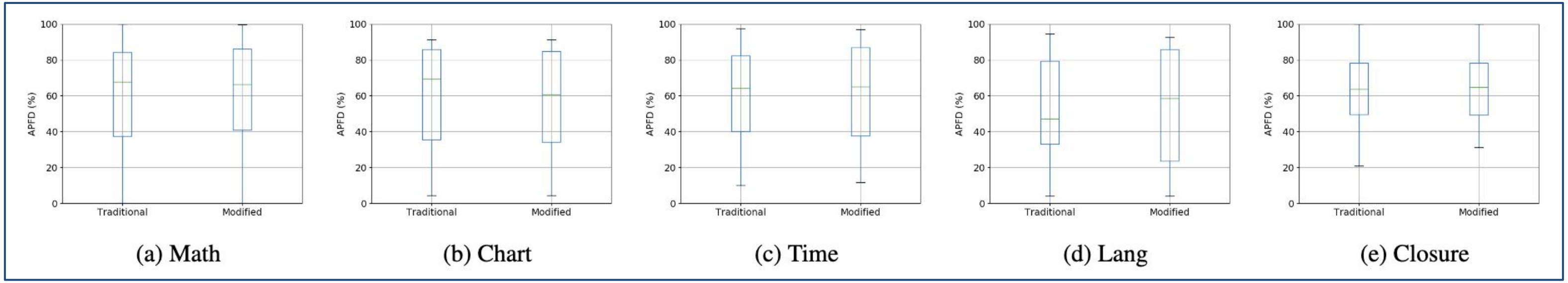
Results

RQ2: Evaluation Results of APFD Scores of Total Strategy (Traditional vs Modified)

Our Results



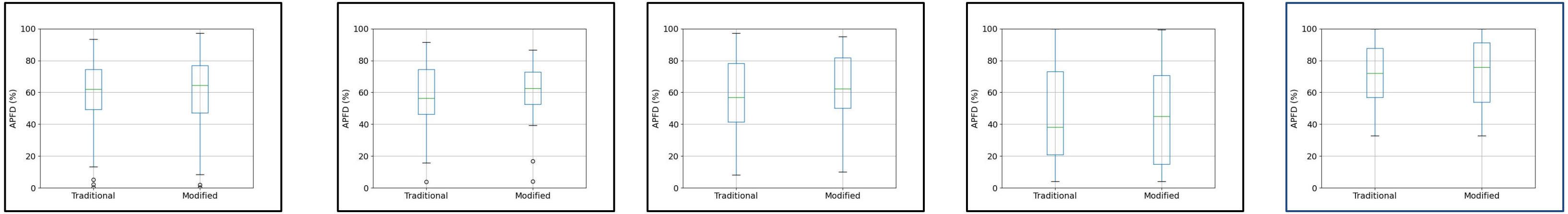
Their Results



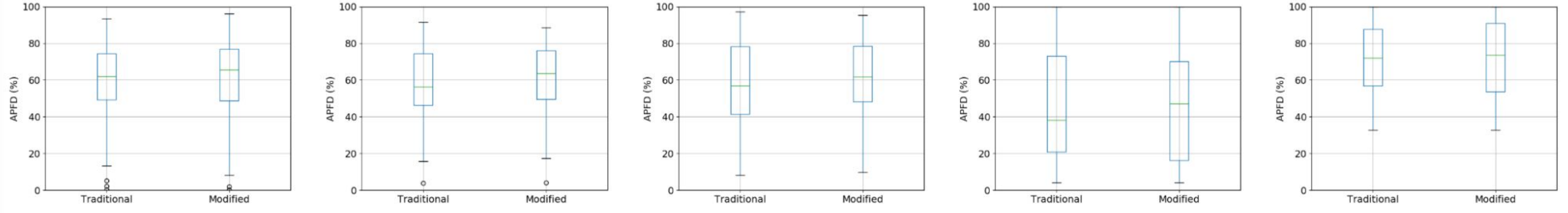
(a) Math (b) Chart (c) Time (d) Lang (e) Closure

RQ1: Evaluation Results of APFD Scores of Additional Strategy (Traditional vs Modified)

Our Results



Their Results



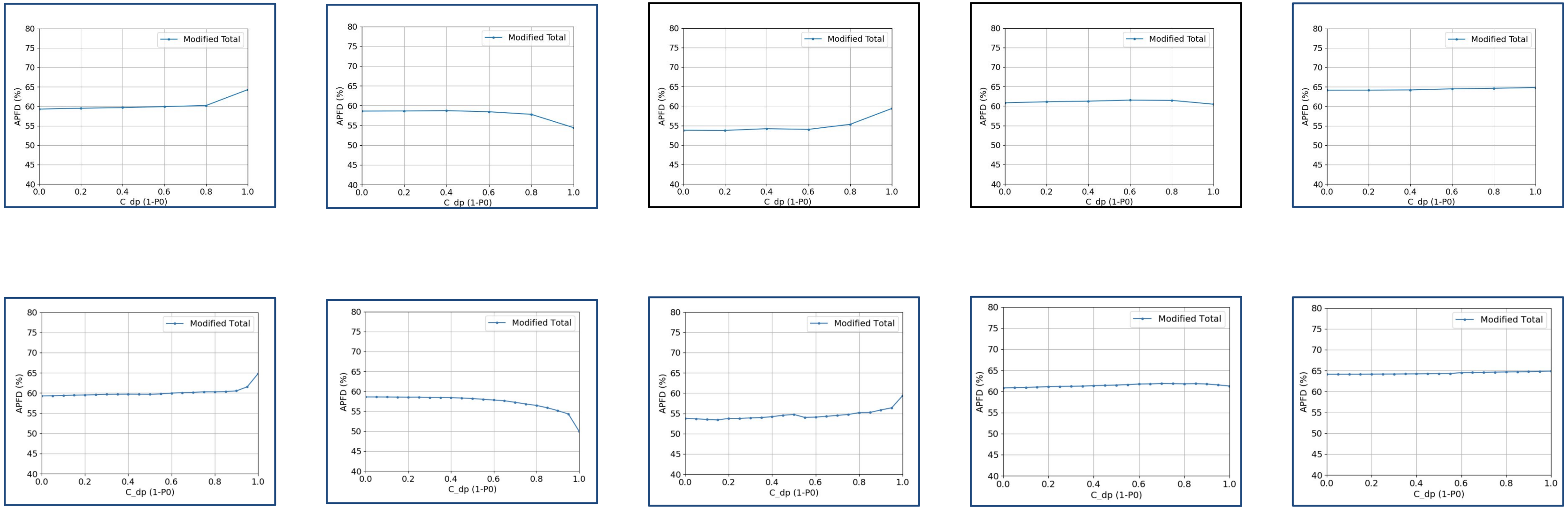
(a) Math (b) Chart (c) Time (d) Lang (e) Closure

RQ2: Mean APFD scores

Library	**Additional Strategy**			**Total Strategy**		
	Traditional	Modify	Changes	Traditional	Modify	Changes
Chart	55.47%	**57.84%**	5.95%	**58.59%**	57.25%	-2.68%
Closure	**71.06%**	71.01%	-0.14%	64.13%	**64.59%**	1.44%
Lang	45.76%	**46.92%**	7.41%	53.80%	**54.51%**	6.95%
Math	59.43%	**60.78%**	4.37%	60.84%	**61.88%**	9.32%

Overall, **mean APFD Scores** of Modified strategies were **higher** than Traditional strategies.

RQ3: Relationship between APFD and C_{dp} (Modified Total)



Our Results

Their Results

Time

Chart

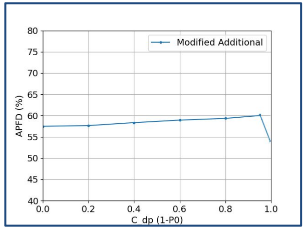
Lang

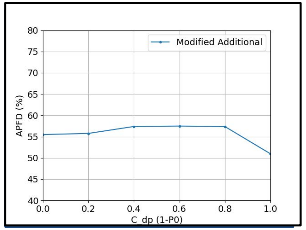
Math

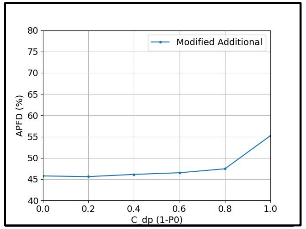
Closure

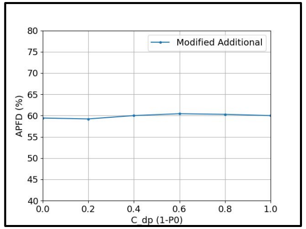
RQ3: Relationship between APFD and C_{dp} (Modified Additional)

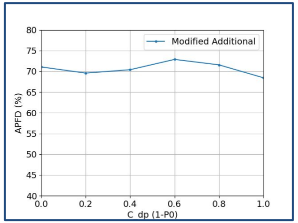
Our Results



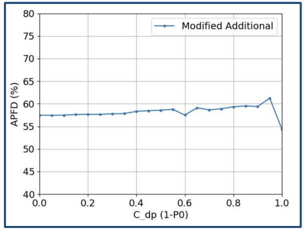


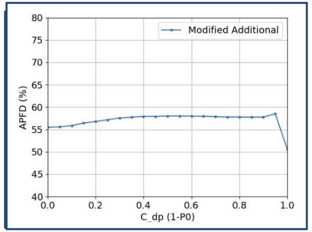


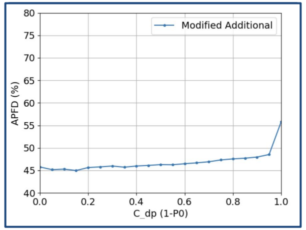


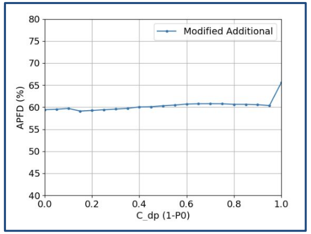


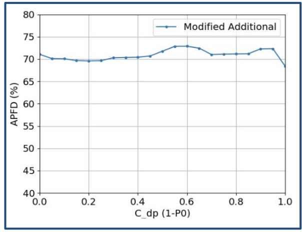
Their Results





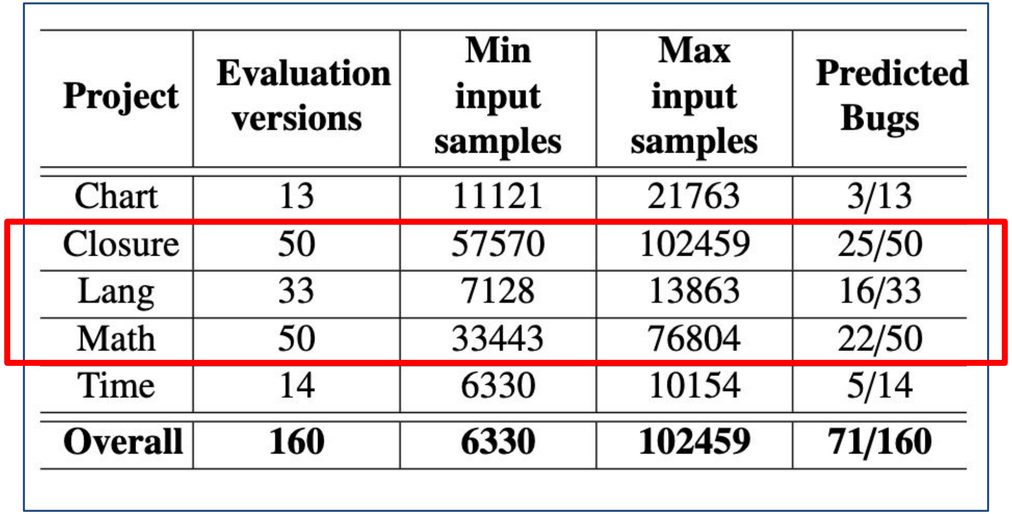




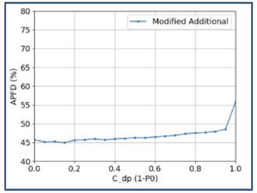


Time **Chart** **Lang** **Math** **Closure**

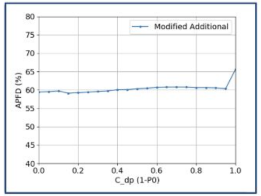
RQ3: Relationship between APFD and C_{dp} (Modified Additional)



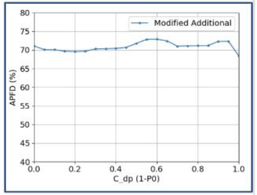
Project	Evaluation versions	Min input samples	Max input samples	Predicted Bugs
Chart	13	11121	21763	3/13
Closure	50	57570	102459	25/50
Lang	33	7128	13863	16/33
Math	50	33443	76804	22/50
Time	14	6330	10154	5/14
Overall	**160**	**6330**	**102459**	**71/160**



Lang



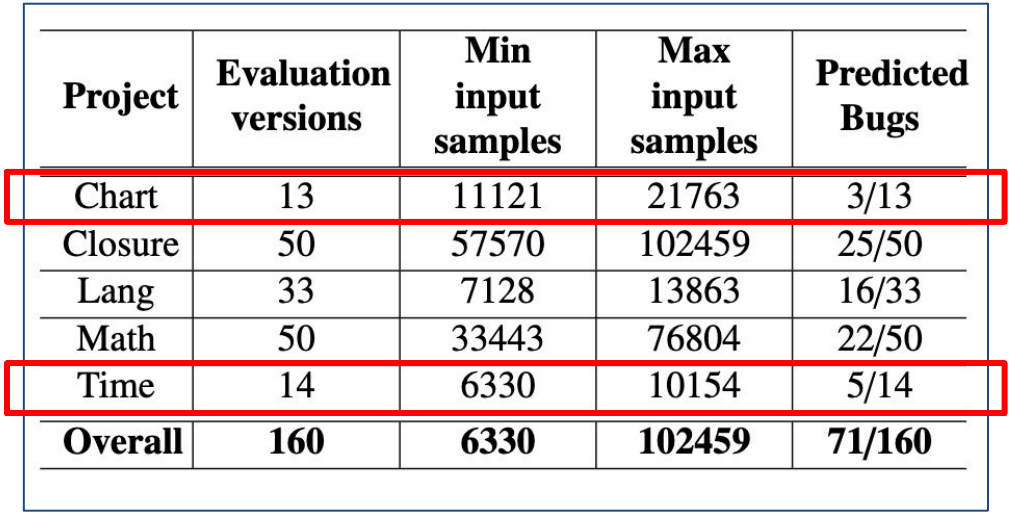
Math



Closure

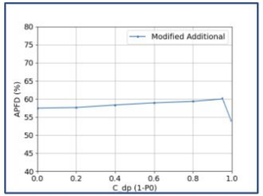
Overall, **modified strategy** performs **better** when the **fault-proneness score** is more taken into account

RQ3: Relationship between APFD and C_{dp} (Modified Additional)

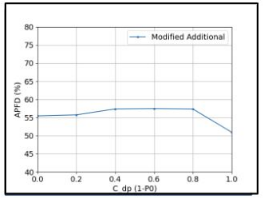


Project	Evaluation versions	Min input samples	Max input samples	Predicted Bugs
Chart	13	11121	21763	3/13
Closure	50	57570	102459	25/50
Lang	33	7128	13863	16/33
Math	50	33443	76804	22/50
Time	14	6330	10154	5/14
Overall	**160**	**6330**	**102459**	**71/160**

Our Results



Time



Chart

Lower number of bugs from smaller libraries cause an inaccurate bug prediction model

Key Takeaways

- **Modified Strategies:** The modified strategies (e.g., incorporating fault-proneness) significantly outperform traditional alternatives, consistent with the paper's conclusion that clustering-based methods improve fault detection rates by leveraging fault-proneness estimations

- **Alignment with Authors:** Our findings closely align with the conclusions of Mahdieh et al. (2020), validating their claims about the superiority of incorporating fault-proneness and clustering in test case prioritization

- **Library Size Dependency:** However, we found out that a larger library size is crucial for accurate and sufficient defect prediction outputs. Smaller projects (e.g., Chart) yield less reliable predictions due to limited data, mirroring the challenges identified in the original study

Future Work

- **Impact of Library Size:** Conduct a comprehensive analysis of test prioritization algorithms across libraries of varying sizes to better understand scalability and performance differences in diverse contexts.

- **Cross-Project Defect Prediction:** Investigate the integration of cross-project defect prediction techniques to enhance the robustness and applicability of fault-proneness estimations across different projects.

- **Comparative Analysis of Modified Algorithms:** Evaluate how the modified versions of various other test prioritization algorithms perform relative to their traditional counterparts.