Enhancing Genetic Improvement of Software with Regression Test Selection

Giovani Guizzo*, Justyna Petke*, Federica Sarro* and Mark Harman*^ *Department of Computer Science, University College London (UCL), London, United Kingdom ^Facebook, London, United Kingdom
{g.guizzo, j.petke, f.sarro, mark.harman}@ucl.ac.uk

Presented By,

Mani Prathi, Anushka Aravelli , Twinkle Reddy

Introduction:

- Genetic Improvement (GI) represents an innovative approach in software engineering, leveraging Artificial Intelligence (AI) techniques to iteratively enhance existing software properties.
- The main difficulty with GI is that it requires a great deal of computational time to conduct the extensive testing necessary to evaluate various software variations.
- This paper presents Regression Test Selection (RTS), a strategic solution meant to reduce testing overhead and maximize the GI process's efficiency.

Genetic Improvement:

Software

Property Improvement

Genetic Modifications

Test Suite

Problem addressed:

- The iterative nature of Genetic Improvement (GI) requires thorough testing of multiple software variants, a process inherently time-consuming and resource-intensive.
- Despite the potential benefits of GI, its practical applicability is hindered by the prolonged execution time, particularly in industrial contexts where efficiency is paramount.
- Addressing the challenge of reducing computational time while maintaining software quality is critical for realizing the full potential of GI and its widespread adoption.

Motivation

- The primary motivation stems from the need to reduce computational time and resource expenditure associated with the GI process.
- Efficient methods are essential for accelerating software improvement processes, enabling rapid iterations and timely feedback.
- To enhance the efficacy and applicability of GI in real-world scenarios, state-of-the-art techniques must be integrated with established software engineering practices.

Proposed solution

The proposed solution involves the integration of Regression Test Selection (RTS) techniques into the Genetic Improvement (GI) framework.
Both dynamic and static RTS methods are employed to selectively execute pertinent test cases, thereby reducing testing overhead and computational costs.

RTS

Cover the changed code

Faster Execution

As a result of incorporating RTS into GI, substantial efficiency improvements can be achieved while maintaining software quality and functionalities.

Combine

Effectiveness

Efficiency

Trade Off

Research Questions:

- **RQ1.** Effectiveness:

  How effective is regression test selection in the context of Genetic Improvement of software?

- **RQ2.** Efficiency:

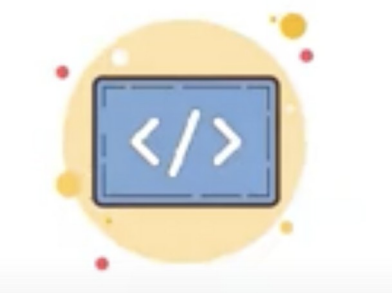
  What is the efficiency gain when using Regression Test Selection with Genetic Improvement?

- **RQ3.** Trade-Off:

  What is the trade-off between efficiency and efficacy of the Genetic Improvement process with various Regression Test Selection strategies in different application scenarios?

Experimental Design

Gin Tool
https://github.com/gintool/gin



Dataset with 7 real world program's codec-1.14 ,compress-1.20 ,csv-1.7 ,fileupload-1.4 ,imaging-1.0,text-1.3,validator-1.6

Comparison:

1.GI- GI with no RTS.
2. GI+Random - randomly selects a subset of test cases
without guidance.
3. GI+Ekstazi - GI using Ekstazi as a dynamic analysis
RTS technique.
4. GI+STARTS - GI using STARTS as a static analysis
RTS technique.

Experimental procedure

- Profiling – selection is done here – save execution time;
- GI execution – save execution time;
- Test against the whole test suite – save test results;
- Compute metrics.

TABLE I

SUBJECT PROGRAMS. LLOC: NUMBER OF LOGICAL LINES OF CODE (EXECUTABLE LINES); #T: NUMBER OF TEST CASES IN THE PROGRAM'S TEST SUITE; T. LLOC: NUMBER OF LOGICAL LINES OF TEST CODE; COV: STATEMENT AND BRANCH COVERAGE PERCENTAGES OBTAINED BY THE TEST SUITE; TEST TIME: EXECUTION TIME OF THE TEST SUITE (MM:SS).

Program	LLOC	#T	T. LLOC	Cov	Test Time
codec-1.14	9 044	1 081	13 276	96/91	00:15
compress-1.20	25 978	1 170	22 059	84/75	01:39
csv-1.7	1 845	325	4 864	89/85	00:06
fileupload-1.4	2 425	82	2 284	80/76	00:04
imaging-1.0	31 320	583	7 427	73/59	00:52
text-1.3	8 703	898	12 872	97/96	00:05
validator-1.6	7 409	536	8 352	86/76	00:11

Answers to RQ1- Effectiveness

- GI+Random : 5 invalid patches (out of 140)
- GI+Starts: 1 invalid patches (out of 140)
- GI+Ekstazi: no invalid patch
- RTS strategies are proven to be safe and effective with GI, ensuring nearly 100% patch validity and significant improvement quality enhancements.

TABLE IV

RQ1. MEDIAN RELATIVE IMPROVEMENT CHANGE (RIC) OF STRATEGIES. GREATER VALUES ARE BETTER. BEST RIC VALUES OR VALUES STATISTICALLY EQUIVALENT TO THE BEST ONES ARE HIGHLIGHTED IN BOLD (P-VALUES < 0.05).

Program	GI	+Ekstazi	+STARTS	+Random	p-value
codec	1	**3.36**	**2.29**	0.95	**1.10e-06**
compress	1	**2.53**	**5.81**	**3.96**	**2.07e-07**
csv	1	**2.89**	**3.8**	**1.68**	**7.10e-04**
fileupload	**1**	**2.55**	**1.97**	**2.08**	6.50e-02
imaging	**1**	**0.64**	**0.95**	**0.46**	0.55240
text	1	**2.40**	0.95	0.57	**2.70e-03**
validator	1	**4.92**	**2.81**	**4.04**	**1.90e-04**
Median	1	**2.55**	2.29	1.68	–

TABLE V

RQ1. EFFECT SIZES FOR THE RELATIVE IMPROVEMENT CHANGE (RIC). EFFECT SIZES GREATER THAN 0.5 MEAN POSITIVE IMPROVEMENT FOR THE LEFT STRATEGY. DIFFERENCES: N = NEGLIGIBLE, S = SMALL, M = MEDIUM, AND L = LARGE. LARGE EFFECT SIZES ARE HIGHLIGHTED IN BOLD.

Program	GI/+Ekstazi	GI/+STARTS	+Ekstazi/+STARTS
codec	**0.04 (L)**	**0.14 (L)**	0.56 (S)
compress	**0.13 (L)**	**0.005 (L)**	0.3 (M)
csv	**0.16 (L)**	**0.2 (L)**	0.49 (N)
fileupload	**0.26 (L)**	0.4 (S)	0.63 (S)
imaging	0.58 (S)	0.48 (N)	0.48 (N)
text	**0.29 (L)**	0.56 (S)	**0.73 (L)**
validator	**0.11 (L)**	**0.22 (L)**	0.66 (M)

Answers to RQ2- Efficiency

- RTS strategies like Ekstazi and STARTS reduce GI execution time costs by up to 68%, with an average savings of 22%.
- 86% of programs showed significant time savings with RTS, with 71% demonstrating a statistically large difference.
- Efficiency varies based on project specifics and test case costs, not merely the number of tests.
- RTS strategies cut down computational time significantly, saving from hours to over a day for programs with expensive tests.
- Using RTS with GI reduces total experimentation time by over a third, proving its effectiveness and environmental benefits.

TO THE BEST ONES) ARE HIGHLIGHTED IN BOLD (P-VALUES < 0.05).

Program	GI	+Ekstazi	+STARTS	+Random	p-value
codec	1.00	**0.39**	**0.40**	0.77	**3.11e-09**
compress	1.00	**0.32**	**0.38**	0.89	**2.12e-12**
csv	1.00	1.03	**0.93**	**0.82**	**7.97e-03**
fileupload	**1.00**	1.11	**0.93**	**1.05**	**2.81e-02**
imaging	1.00	0.79	0.98	**0.40**	**6.48e-09**
text	1.00	**0.67**	**0.78**	1.15	**2.60e-05**
validator	1.00	**0.82**	**0.47**	1.01	**1.50e-02**
Median	1.00	0.79	**0.78**	0.89	–

TABLE VII

RQ2. EFFECT SIZES FOR THE RELATIVE COST (RC). EFFECT SIZES GREATER THAN 0.5 MEAN GREATER COST FOR THE LEFT STRATEGY. DIFFERENCES: N = NEGLIGIBLE, S = SMALL, M = MEDIUM, AND L = LARGE. LARGE EFFECT SIZES ARE HIGHLIGHTED IN BOLD.

Program	GI/+Ekstazi	GI/+STARTS	+Ekstazi/+STARTS
codec	**0.95 (L)**	**0.89 (L)**	0.55 (N)
compress	**0.99 (L)**	**0.98 (L)**	0.30 (M)
csv	0.49 (N)	0.65 (M)	0.70 (M)
fileupload	0.35 (S)	0.63 (S)	**0.74 (L)**
imaging	**0.80 (L)**	0.52 (N)	**0.24 (L)**
text	**0.82 (L)**	**0.76 (L)**	0.35 (S)
validator	0.63 (S)	**0.73 (L)**	0.64 (M)

RQ2- Efficiency

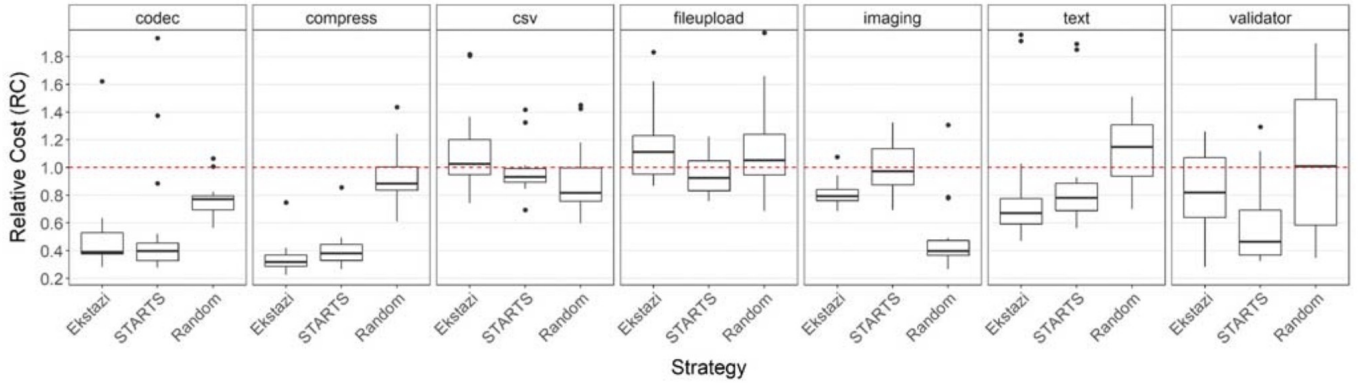


Fig. 1. **RQ2. Relative Cost (RC) of strategies.** Lower values are better. The dashed line represents the median cost of GI without RTS.

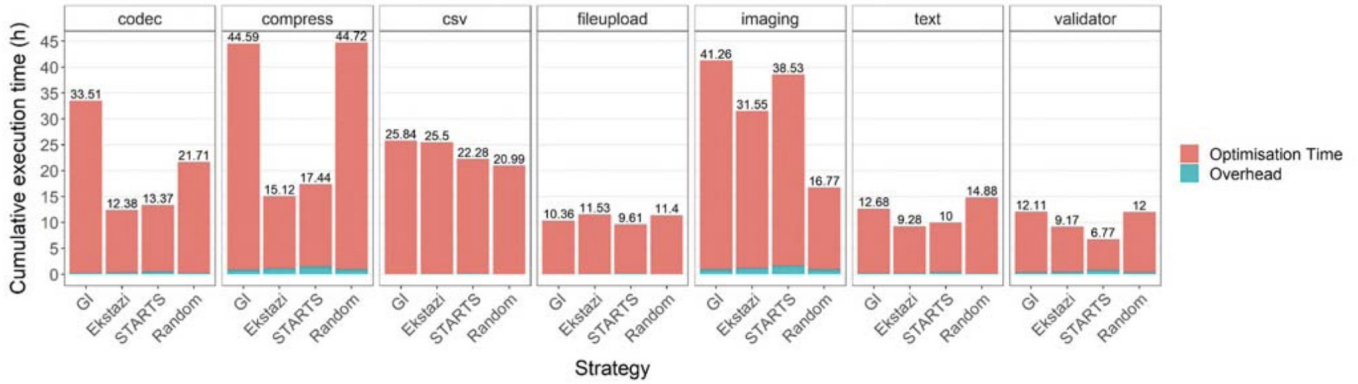


Fig. 2. **RQ2. Cumulative execution times** Results are for four GI strategies i) without RTS, ii) and iii) with RTS, and iv) GI with random test selection.

Answers to RQ3- Trade-Off

P_{improv} **USE CASE.** MEDIAN IMPROVEMENT FOUND IN SECONDS. GREATER VALUES ARE BETTER.

Program	GI	+Ekstazi	+STARTS
codec	1.29	**4.35**	2.96
compress	2.46	6.23	**14.28**
csv	0.82	2.25	**2.95**
fileupload	0.05	**0.13**	0.11
imaging	11.98	7.63	**12.62**
text	1.46	**3.52**	1.40
validator	0.29	**1.44**	0.82
Median	1.29	**3.52**	2.95

RQ3. F_{improv} **USE CASE.** MEDIAN TIME IN SECONDS EAC TOOK TO FIND A POSITIVE AND VALID IMPROVEMENT. LO ARE BETTER.

Program	GI	+Ekstazi	+STARTS
codec	711.32	215.76	**188.19**
compress	730.20	251.14	**244.01**
csv	**388.12**	474.82	425.26
fileupload	287.02	**261.31**	354.38
imaging	768.03	**562.79**	659.11
text	237.69	137.69	**137.38**
validator	131.78	**79.96**	142.68
Median	388.12	251.14	**244.01**

- GI+Ekstazi leads in achieving the highest median improvement, making it the optimal choice for seeking the best valid software improvements.
- GI+STARTS excels in speed, finding the first positive improvement fastest, averaging 244 seconds, and is preferred for rapid enhancement.
- GI+Ekstazi demonstrates superiority in generating a wider variety of valid and positive improvements across multiple programs.
- For scenarios prioritizing quality or diversity of improvements, GI+Ekstazi is recommended; for speed, GI+STARTS is the choice.
- GI combined with RTS strategies shows superior results in most cases compared to traditional GI, recommending RTS integration in future GI work.

Novelty

- First to Explore RTS for Non-Functional Improvement in GI.
- Leveraging strengths of GI and RTS for enhanced efficiency and effectiveness
- Functional and Non-functional Focus: Addresses both types of software improvement, unlike previous APR-centric studies, broadening the scope of application.
- Comprehensive Evaluation with Real-World Java Programs.

Assumptions

- The study assumes that the behavior and performance of the selected RTS techniques (Ekstazi and STARTS) remain stable across different software projects and experimental conditions.
- The chosen parameter configurations are equally suitable for all projects, which may not be the case in practice.
- The study assumes a static composition of the test suites during the experimentation process, with no changes to the test cases or test coverage throughout the optimization process.

Limitations

- Selection of software projects may not accurately represent the broader population.
- Analysis limited to two state-of-the-art Regression Testing Selection (RTS) techniques.
- potential biases due to experimental setup and execution conditions.
- Optimization algorithms can have unpredictable running times because they involve random processes.

Practical significance of proposed solution:

- **Streamlined Development Processes**:
  - Integration of RTS techniques into CI/CD pipelines accelerates development cycles.
  - Enables faster iteration and deployment of software updates.
- **Enhanced Quality Assurance Practices**:
  - RTS ensures thorough testing while minimizing redundant test executions.
  - Improves software reliability and stability by focusing on relevant test cases.
- **Efficiency Gains in Software Improvement**:
  - Combining RTS with GI significantly reduces the computational time required for fitness evaluation.
  - Offers up to 68% efficiency gain, making GI more feasible for larger projects.
- **Potential for Cross-Domain Application**:
  - Provides a framework for addressing diverse challenges and enhancing automation processes.

DISCUSSION POINTS

- What are other test case selection strategies could be investigated.
- What characteristics a given test suite should have in order to be effective using GI process.
- Does the approach still work if the software is very complex or if there aren’t many tests available?

Thank You !!