# EMPIRICAL EVALUATION OF THE TARANTULA AUTOMATIC FAULT-LOCALIZATION TECHNIQUE

AUTHORS:

*JAMES A. JONES*
*&*
*MARY JEAN HARROLD*

# Introduction

- Fault localization is the process of identifying the faulty parts of a program by analyzing the results of executed test cases.

- The goal is to reduce the time it takes for developers to identify and fix bugs by using automated tools.

- Several techniques are used for fault localization, including statistical methods like Tarantula.

# What is Tarantula?

- Tarantula is a fault-localization tool that uses test results to rank program statements based on how likely they are to contain errors.

- Statements run more often by failed test cases are considered more suspicious.

- **Visualization:**
  - Red: High suspicion of error.
  - Yellow: Medium suspicion.
  - Green: Low suspicion.

$$hue(s) = \frac{\frac{passed(s)}{totalpassed}}{\frac{passed(s)}{totalpassed} + \frac{failed(s)}{totalfailed}}$$

# How Tarantula works?

- Tarantula uses test results to calculate how suspicious each line of code is.
- This simple code takes three numbers as input and prints the middle number. There is a bug at line 7: the line should assign m = x; instead of m = y;

$$suspiciousness(e) = 1 - hue(e) = \frac{\frac{failed(e)}{totalfailed}}{\frac{passed(e)}{totalpassed} + \frac{failed(e)}{totalfailed}}$$

| | | Test Cases | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | mid() {<br>int x,y,z,m; | 3,3,5 | 1,2,3 | 3,2,1 | 5,5,5 | 5,3,4 | 2,1,3 | suspiciousness | rank |
| 1: | read("Enter 3 numbers:",x,y,z); | ● | ● | ● | ● | ● | ● | 0.5 | 7 |
| 2: | m = z; | ● | ● | ● | ● | ● | ● | 0.5 | 7 |
| 3: | if (y<z) | ● | ● | ● | ● | ● | ● | 0.5 | 7 |
| 4: | if (x<y) | ● | ● | | | ● | ● | 0.63 | 3 |
| 5: | m = y; | | ● | | | | | 0.0 | 13 |
| 6: | else if (x<z) | ● | | | | ● | ● | 0.71 | 2 |
| 7: | m = y; // *** bug *** | ● | | | | | ● | 0.83 | 1 |
| 8: | else | | | ● | ● | | | 0.0 | 13 |
| 9: | if (x>y) | | | ● | ● | | | 0.0 | 13 |
| 10: | m = y; | | | ● | | | | 0.0 | 13 |
| 11: | else if (x>z) | | | | ● | | | 0.0 | 13 |
| 12: | m = x; | | | | | | | 0.0 | 13 |
| 13: | print("Middle number is:",m); | ● | ● | ● | ● | ● | ● | 0.5 | 7 |
| | } Pass/Fail Status | P | P | P | P | P | F | | |

**Figure 1: Example of Tarantula technique.**

# Empirical Study

- **Objective:** Compare Tarantula to four other fault-localization techniques.

- **Techniques Compared:**
  - Set Union
  - Set Intersection
  - Nearest Neighbor
  - Cause Transitions

- **Evaluation Criteria:**
  - Effectiveness: How accurately each technique locates faults.
  - Efficiency: The speed at which each technique performs fault localization.

# Experimental Setup

- Programs Used: Siemens suite, consisting of 7 programs with 132 faulty versions in total.
- Includes programs like lexical analyzers, schedulers, and replacements.
- Test Cases: Designed to cover all paths within the programs to thoroughly test their functionality.

**Table 1: Objects of Analysis**

| Program | Faulty Versions | Procedures | LOC | Test Cases | Description |
|---|---|---|---|---|---|
| print_tokens | 7 | 20 | 472 | 4056 | lexical analyzer |
| print_tokens2 | 10 | 21 | 399 | 4071 | lexical analyzer |
| replace | 32 | 21 | 512 | 5542 | pattern replacement |
| schedule | 9 | 18 | 292 | 2650 | priority scheduler |
| schedule2 | 10 | 16 | 301 | 2680 | priority scheduler |
| tcas | 41 | 8 | 141 | 1578 | altitude separation |
| tot_info | 23 | 16 | 440 | 1054 | information measure |

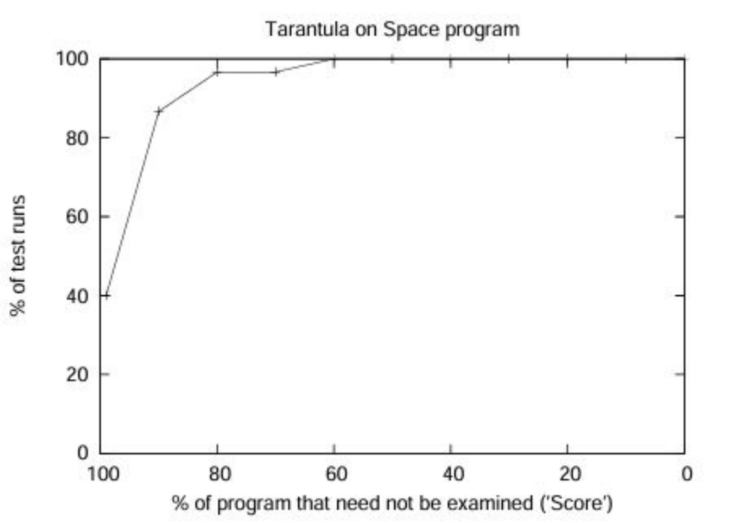
# Fault-Localization Techniques Comparison

- **Set Union:** Compares the difference between the test coverage of passed and failed tests to identify suspicious code.
- **Set Intersection:** Looks at the intersection of statements executed by all passed tests but not by failed tests to detect faults.
- **Nearest Neighbor:** Identifies the passed test that most resembles the failed test and eliminates matching statements to focus on faults.
- **Cause Transitions:** Focuses on memory state transitions to pinpoint likely locations of faults.

# Dependent and Independent Variable

- Independent Variable: Fault-localization technique used (Tarantula, Set Union, Set Intersection, Nearest Neighbor, Cause Transitions).

- Dependent Variables:
    - Effectiveness: Percentage of code not examined before the fault was found.
    - Efficiency: Time taken for computation and I/O processes.



# Effectiveness comparison by Score

- Tarantula's Performance:
- At 99% score level: Tarantula pinpointed faults in 13.9% of runs.
- At 90-99% score level: Tarantula located faults in 41.8% of runs.
- Graph: % of runs vs. % of program not examined (showing comparison between Tarantula and other techniques).

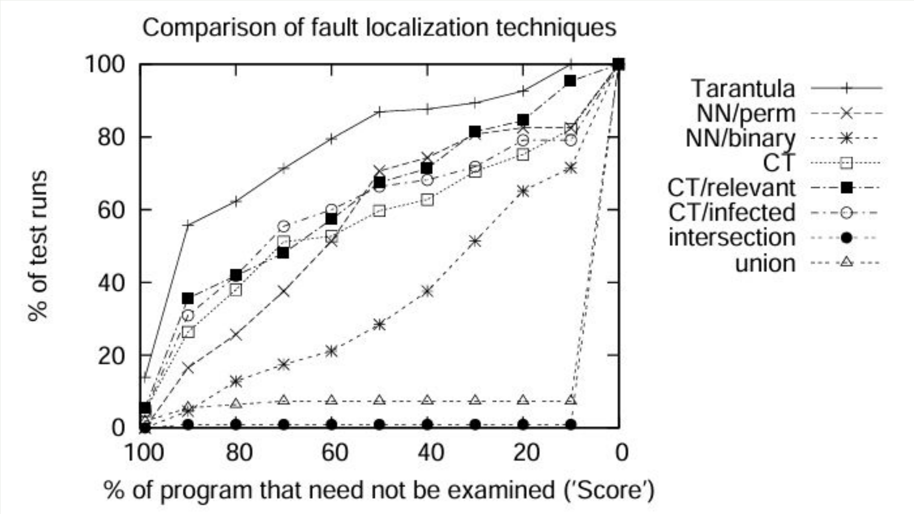
**Table 2: Percentage of test runs at each score level.**

| Score | Tarantula | NN/perm | NN/binary | CT | CT/relevant | CT/infected | Intersection | Union |
|---|---|---|---|---|---|---|---|---|
| 99-100% | 13.93 | 0.00 | 0.00 | 4.65 | 5.43 | 4.55 | 0.00 | 1.83 |
| 90-99% | 41.80 | 16.51 | 4.59 | 21.71 | 30.23 | 26.36 | 0.92 | 3.67 |
| 80-90% | 5.74 | 9.17 | 8.26 | 11.63 | 6.20 | 10.91 | 0.00 | 0.92 |
| 70-80% | 9.84 | 11.93 | 4.59 | 13.18 | 6.20 | 13.64 | 0.00 | 0.92 |
| 60-70% | 8.20 | 13.76 | 3.67 | 1.55 | 9.30 | 4.55 | 0.00 | 0.00 |
| 50-60% | 7.38 | 19.27 | 7.33 | 6.98 | 10.08 | 6.36 | 0.00 | 0.00 |
| 40-50% | 0.82 | 3.67 | 9.17 | 3.10 | 3.88 | 1.82 | 0.00 | 0.00 |
| 30-40% | 0.82 | 6.42 | 13.76 | 7.75 | 10.08 | 3.64 | 0.00 | 0.00 |
| 20-30% | 4.10 | 1.83 | 13.76 | 4.65 | 3.10 | 7.27 | 0.00 | 0.00 |
| 10-20% | 7.38 | 0.00 | 6.42 | 6.98 | 10.85 | 0.00 | 0.00 | 0.00 |
| 0-10% | 0.00 | 17.43 | 28.44 | 17.83 | 4.65 | 20.91 | 99.08 | 92.66 |

# Effectiveness Graph

- Visual representation comparing Tarantula with other techniques across various score levels.

- Tarantula has the highest percentage of high scores in identifying faults.



# Efficiency comparison

- Tarantula Computation Time: 0.0025–0.0063 seconds.
- Cause Transitions Time: 180–6500 seconds.
- Tarantula's Efficiency Advantage: Two orders of magnitude faster than Cause Transitions.

| Program | Tarantula (computation only) | Tarantula (including I/O) | Cause Transitions |
|---|---|---|---|
| print_tokens | 0.0040 | 68.96 | 2590.1 |
| print_tokens2 | 0.0037 | 50.50 | 6556.5 |
| replace | 0.0063 | 75.90 | 3588.9 |
| schedule | 0.0032 | 30.07 | 1909.3 |
| schedule2 | 0.0030 | 30.02 | 7741.2 |
| tcas | 0.0025 | 12.37 | 184.8 |
| tot_info | 0.0031 | 8.51 | 521.4 |

# Why Tarantula succeeds

- **Comprehensive Test Data:** Leverages diverse test cases to improve accuracy.
- **Tolerance for Passing Errors:** Doesn't over-penalize statements covered by passing tests.
- **Visualization:** Offers an easy-to-understand, color-coded view to highlight suspicious statements.

# Study Limitations

- **Limited to Single-Fault Scenarios:** Focuses on programs with one fault per version.
- **Siemens Suite Programs:** Small-scale programs, results may not generalize to larger systems.
- **No Real-World User Studies:** Lack of practical user validation for the techniques.

# Summary of Findings

- **Tarantula's Superiority:** Most effective and efficient fault-localization technique tested.
- **Performance:** Outperforms all other techniques in precision and speed.
- **Higher Effectiveness:** At 99% score level, Tarantula pinpointed faults in 13.9% of runs.
- **High Accuracy:** Tarantula found faults in 41.8% of runs at 90-99% score level.
- **Speed Advantage:** Tarantula is two orders of magnitude faster than Cause Transitions.
- **Test Data & Visualization:** Comprehensive use of test data and visual representation aid in identifying suspicious code.
- **Tolerance for Passing Faults:** Allows tolerance for faults in passing tests, improving robustness.

# Thank You!

Jayasurya Pazhani
40289512