Prioritizing Test Suites by Finding Hamiltonian Paths: Preliminary Studies and Initial Results

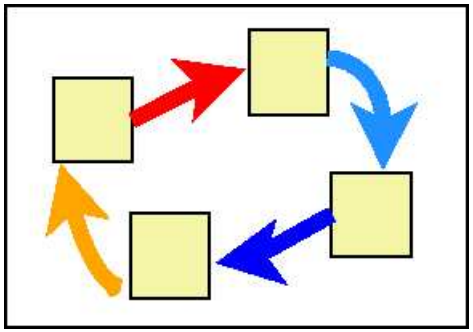
Suvarshi Bhadra and Gregory M. Kapfhammer

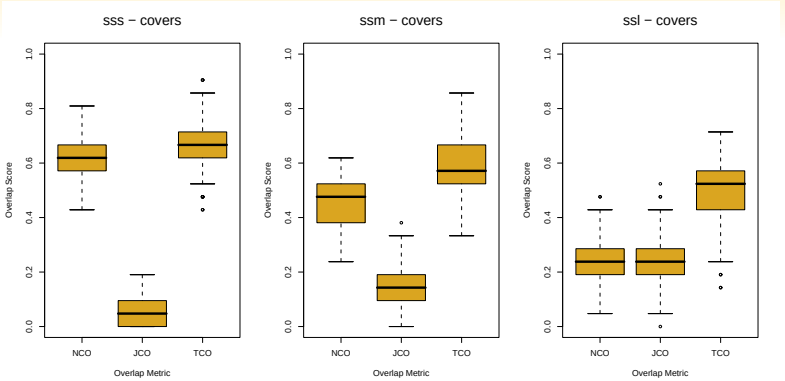
Department of Computer Science
Allegheny College, Pennsylvania, USA
http://www.cs.allegheny.edu/~gkapfham/

TAIC PART 2008, Fast Abstract Track, August 2008

Featuring images from www.pdclipart.org

Important Contributions



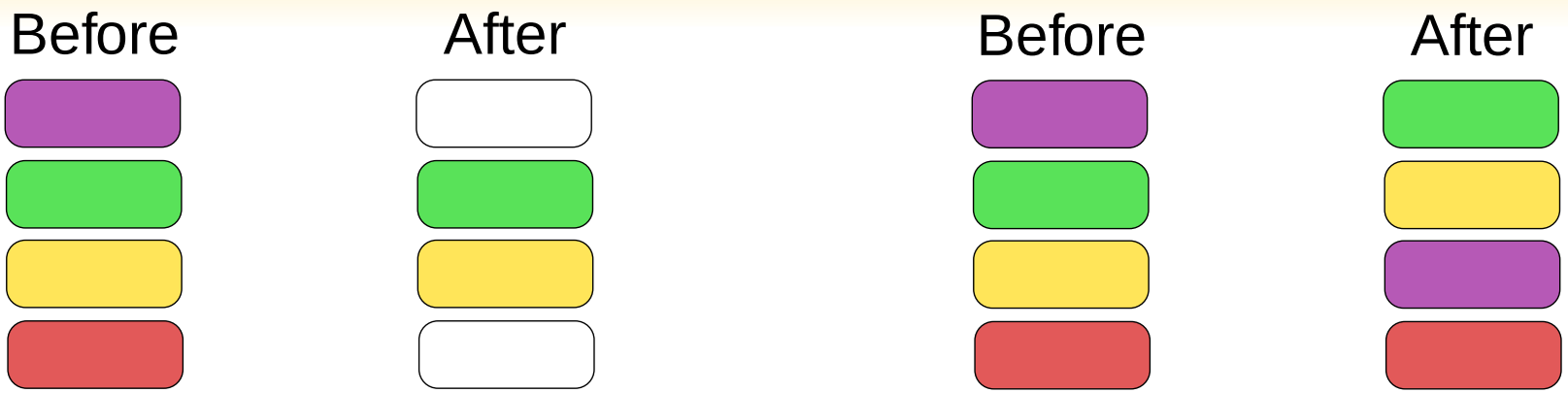


Technique Formulation

Empirical Results

Design, **implement** and empirically **evaluate** test suite prioritizers that leverage travelling salesperson problem (TSP) solvers to **efficiently** find **cost-effective** orderings

Regression Testing Techniques



Reduction Prunes the Test Suite

Prioritization Reorders the Tests

It may **expensive** to run a test suite $T = \langle T_1, \ldots, T_n \rangle$. **Prioritization** searches through the $n! = n \times n - 1 \times \ldots \times 1$ orderings for those that **avoid** costly database **restarts**, Web service **calls**, or memory **interactions**.

Prioritizing When Memory is Constrained

Frequent Memory Rewrites

High Testing Costs

Frequent **reads** and **writes** to memory may **increase** execution time by as much as **600%** when a Java application executes on a virtual machine with a **small heap**

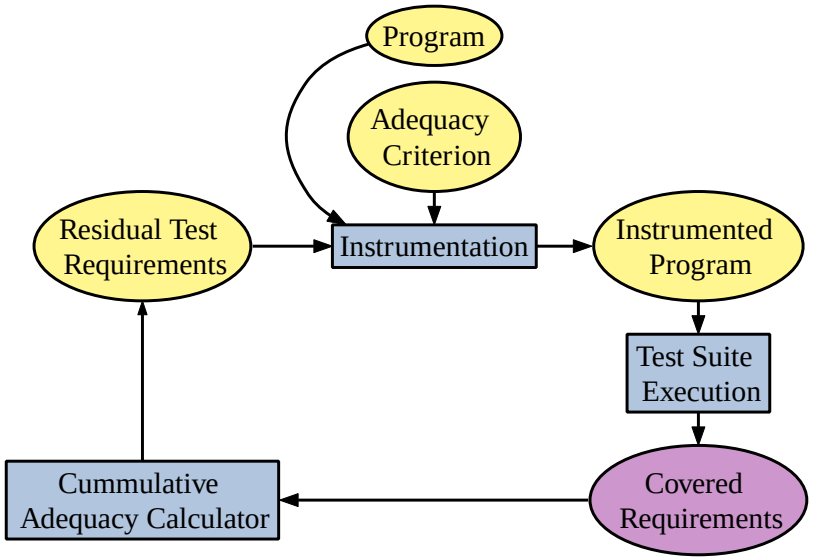
Solution: maximize memory **reuse** between test cases

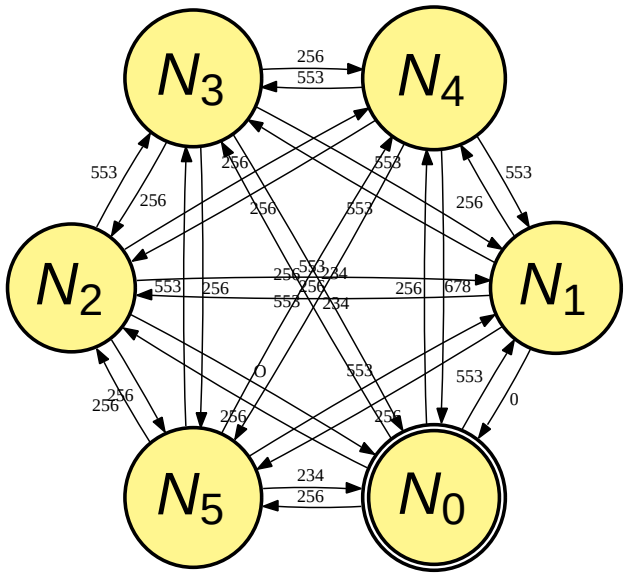
The Impact of Test Ordering

	m_1 30	m_2 30	m_3 30	m_4 30	m_5 30	m_6 30	Test Size
T_1	•	•	•				90
T_2				•	•	•	90
T_3	•	•	•				90
T_4				•	•	•	90
T_5	•	•					60

- $T = \langle T_1, T_2, T_3, T_4, T_5 \rangle$ transfers **750** units to and from memory
- $T' = \langle T_2, T_4, T_1, T_3, T_5 \rangle$ only loads and unloads **180** units

Test Prioritization: Steps One and Two

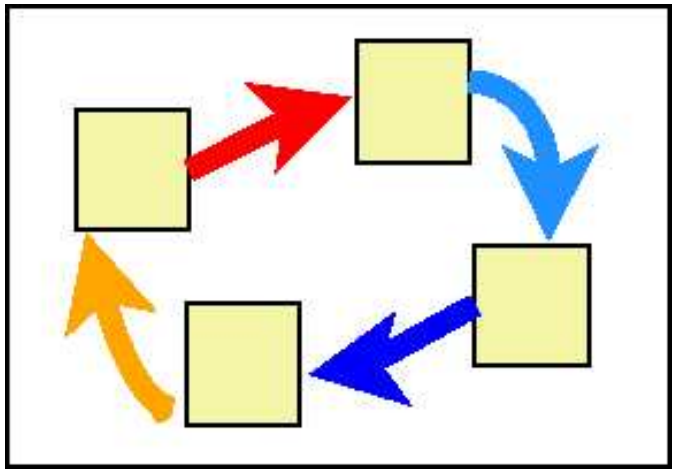


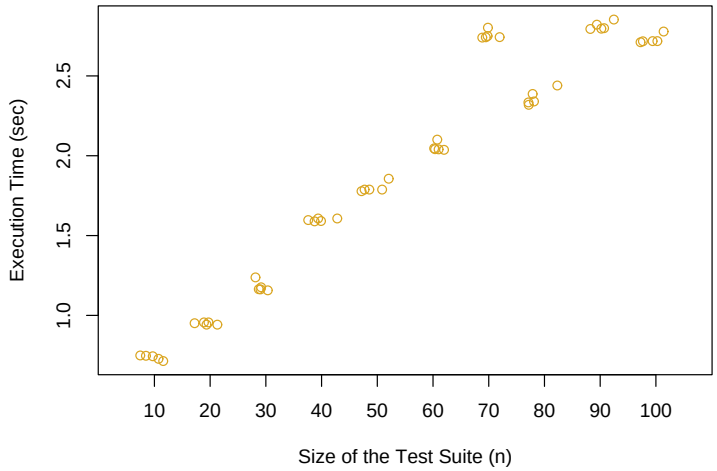


Collect method **invocation** and **size** data using test coverage **monitor** and **profiler**

Formulate a **complete graph** using equations that estimate costs for all **test pairs**

Test Prioritization: Steps Three and Four

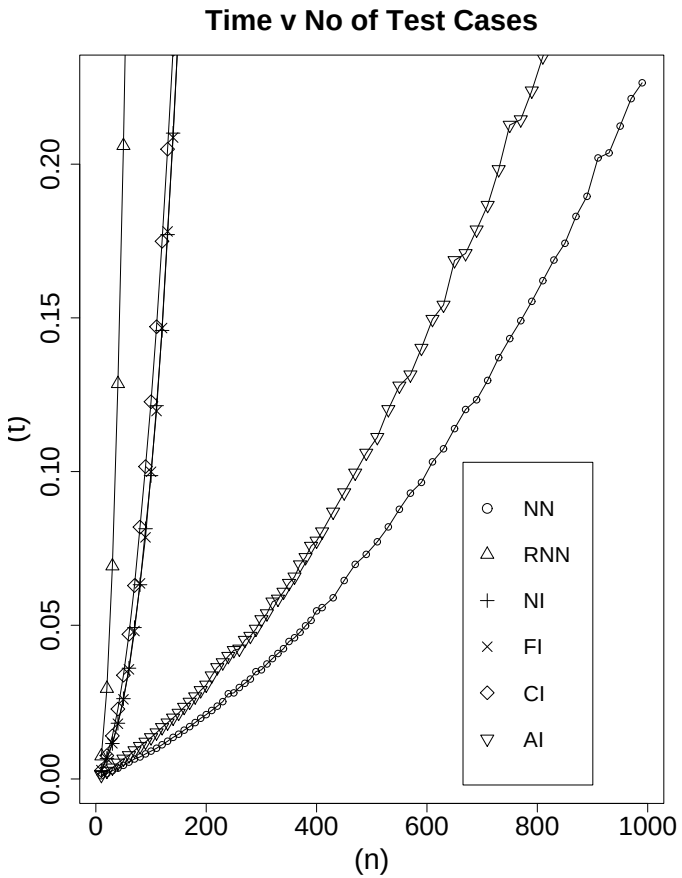


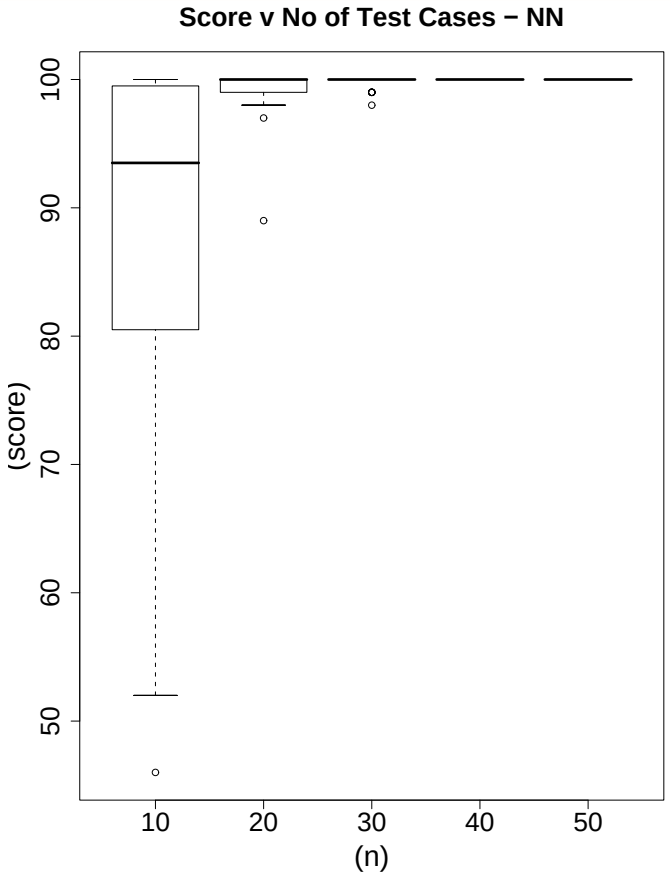


Use **TSP solvers** to identify a **Hamiltonian path** with low estimated costs

Evaluate the efficiency of the TSP **solvers** and the effectiveness of the test **orderings**

Empirical Results

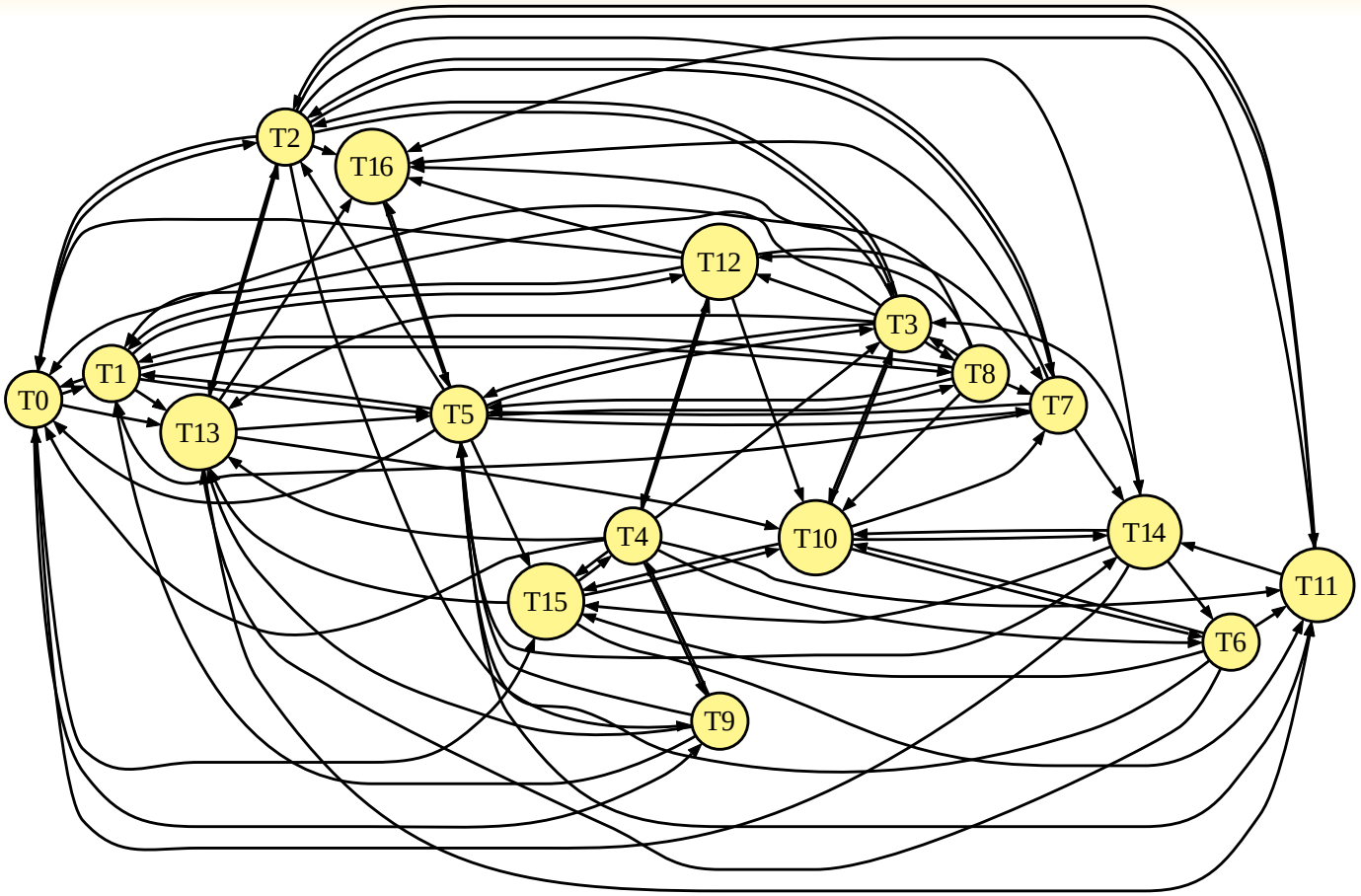




Efficient Prioritizers

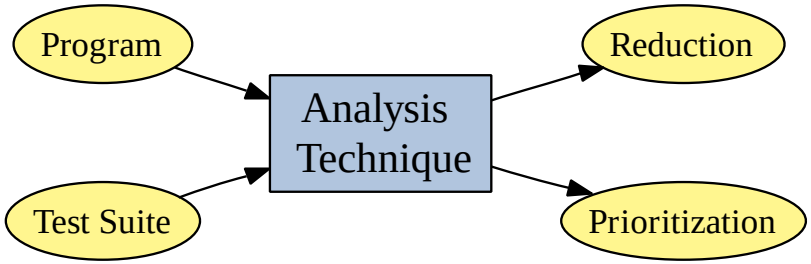
High Percentile Rankings

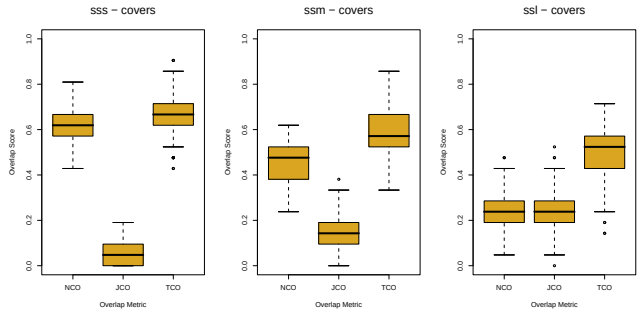
Avoiding Database Restarts



Use **prioritization** to **avoid** costly database **restarts**

Concluding Remarks





Test Suite Prioritizers

Initial Empirical Evaluation

- Preliminary results with synthetic test suites indicate that it is possible to prioritize test suites with TSP solvers
- Use different methods for solving TSP instances (e.g., order-based genetic algorithms) and include real-world applications

http://www.cs.allegheny.edu/~gkapfham/research/