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**

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

**Empirical Results**

- Graphs showing overlap metrics for various prioritizers (NCO, JCO, TCO) and their overlap scores.
Regression Testing Techniques

Before | After
-------|-------
Purple  | White
Green   | Green
Yellow  | Yellow
Red     | Red

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

<table>
<thead>
<tr>
<th></th>
<th>$m_1$</th>
<th>$m_2$</th>
<th>$m_3$</th>
<th>$m_4$</th>
<th>$m_5$</th>
<th>$m_6$</th>
<th>Test Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>$T_2$</td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>$T_3$</td>
<td>•</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>$T_4$</td>
<td></td>
<td></td>
<td></td>
<td>•</td>
<td>•</td>
<td>•</td>
<td>90</td>
</tr>
<tr>
<td>$T_5$</td>
<td>•</td>
<td>•</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

- $T = \langle T_1, T_2, T_3, T_4, T_5 \rangle$ 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
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**

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

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/