Towards Automatic Cost Model Discovery for Combinatorial Interaction Testing

Gulsen Demiroz and Cemal Yilmaz
{gulsend, cyilmaz}@sabanciuniv.edu
Sabanci University, Istanbul, Turkey

5th International Workshop on Combinatorial Testing (IWCT 2016)
April 10, 2016
Combinatorial Interaction Testing (CIT) - A Motivating Example: MySQL -

- A highly configurable system
  - 100+ configuration options
  - Dozens of OS, compiler, and platform combinations

- Assuming each option takes on a binary value
  - \(2^{100+}\) configurations to validate

- Assuming each configuration takes 1 second to test
  - \(2^{100+}\) secs. \(\approx 10^{20+}\) centuries for exhaustive testing
  - Big Bang is estimated to be about \(10^7\) centuries ago

- Exhaustive testing is infeasible!

Which configurations should be tested?
Covering Arrays (CAs)

- Given a coverage strength \( t \) and a configuration space model that includes
  - configuration options
  - their settings
  - inter-option constraints
- A \( t \)-way covering array is a set of configurations, in which each possible combination of option settings for every combination of \( t \) options appears at least once

An example 2-way covering array

<table>
<thead>
<tr>
<th>o1</th>
<th>o2</th>
<th>o3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Basic Justification

(under certain conditions) $t$-way covering arrays can exercise all system behaviors caused by the settings of $t$ or fewer options
To Reduce Testing Cost

standard covering arrays aim to reduce the number of configurations selected by simply assuming that each configuration costs the same
However

*we empirically demonstrated that this assumption does not generally hold true in practice and that testing cost typically varies from one configuration to another*
Example

configuring MySQL with NDB, which enables clustering of in-memory databases, is 50% more expensive than configuring it without NDB.
Unfortunately

when the cost varies, minimizing the number of configurations is not necessarily the same as minimizing actual cost of testing
Solution

take the actual cost of testing into account when constructing covering arrays
Cost-Aware Covering Arrays

- Take as input a configuration space model augmented with a cost function
  - specifying actual cost of testing at the level of option setting combinations
- Compute as output a $t$-way covering array that minimizes the cost function
Example

Assuming that the costs of runtime configurations are negligible compared to those of compile-time configurations and each compile-time configuration costs the same

<table>
<thead>
<tr>
<th>Compile-time</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>o1 o2 o3</td>
<td>o4 o5 o6 o7</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 1 1</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0 1 1 0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>1 0 0 1</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1 1 0 1</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 1 1 1</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

(a) A standard 2-way covering array

<table>
<thead>
<tr>
<th>Compile-time</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>o1 o2 o3</td>
<td>o4 o5 o6 o7</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0 1 1 1</td>
<td>0 0 1 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>1 1 0 0</td>
<td>0 0 0 0</td>
</tr>
</tbody>
</table>

(b) A cost-aware 2-way covering array

Compared to the standard 2-way CA in (a), the cost-aware 2-way CA in (b) reduces the cost by 50% while covering all required combinations
But

*manually specifying the cost function is, in general, cumbersome and error-prone*
Because

- Configuration spaces evolve continuously
- Knowledge about the space is distributed
- Manually defining the cost at the level of option setting combinations is infeasible
- Determining costly combinations is a non-trivial task for developers
- Even if the costly combinations are known, it is hard to express their relative costs in an accurate and precise manner
Discovering Cost Function

- **Input**
  - A standard configuration space model
  - A QA task, the cost function of which will be discovered
  - A means for measuring the cost of carrying out the QA task

- **Approach**
  1. Generate and test a standard \((\geq t+1)\)-way covering array
  2. Use *feature selection* to *identify* combinations of option settings that affect the cost the most
  3. Fit a *generalized linear regression model* to *quantify* the effects of these costly combinations

- **Output**
  - A *cost function* which given a configuration, estimates the cost of carrying out the QA task in the configuration, e.g.,

\[
\text{cost}(c) = 15.14 + 237.15(o_1=1) + 117.42(o_2=2; o_3=3) + \ldots
\]
Experiments

- Subject applications
  - MySQL database server
    - 35 configuration options with varying no of settings
    - 522 test cases
  - Apache web server
    - 40 configuration options with varying no of settings
    - 171 test cases

- QA tasks of interests
  1. Build the system (Task 1)
  2. Run a single test case (Task 2)
  3. Run all test cases (Task 3)

- Cost = the time it takes to carry out the task
Evaluation Framework

- Used 4-way covering arrays for discovery
- Fitted three types of models
  - **Additive**: 1\(^{st}\)-order effects-only models
  - **Non-additive**: 1\(^{st}\)- and 2\(^{nd}\)-order effects models
  - **Significant effects-only**: Only the significant 1\(^{st}\)- and 2\(^{nd}\)-order effects models
- Used the fitted models to predict the costs of randomly generated 2- and 3-way CAs
- \( R^2 \) was used for the evaluations
  - A statistical measure of how close the actual data is to the fitted regression line
  - The higher the \( R^2 \leq 1 \), the better the model is
Summary of Results

- Reliably estimated the costs
  - $R^2 = 0.88$ for MySQL and 0.98 for Apache
- Non-additive models performed better than additive models
  - Additive: $R^2 = 0.79$ for MySQL and 0.97 for Apache
  - Non-additive: $R^2 = 0.92$ for MySQL and 0.98 for Apache
- Significant effects-only models, while greatly reducing the number of terms in the models by 64%, produced comparable results
  - $R^2 = 0.91$ for MySQL and 0.98 for Apache
Future Work

- Design of Experiments (DoE) theory for cost model discovery
- Approaches for generating cost-aware covering arrays
- Cost- and test case-aware CIT
- Cost-aware, feedback driven, adaptive CIT


This research is supported by the Scientific and Technological Research Council of Turkey (113E546)