Using Artificial Life Techniques to Generate Test Cases for Combinatorial Testing

Presentation: TheinLai Wong
Authors: T. Shiba, T. Tsuchiya, T. Kikuno
Osaka University
Backgrounds

• Testing is an important but cost-consuming process.
  – An efficient and cost-effective methodology is required.

• **Combinatorial Testing** has received recent attention.
Combinatorial Testing

- Covers all combinations of values for any $k$ of the input parameters.
  - Rationale: Many faults are caused by interactions of a small number of parameters.

Example

$k = 2$, All parameters have three values.

Number of Tests

$3^4 = 81 \quad \rightarrow \quad 9$

Exhaustive Testing  Combinatorial Testing

<table>
<thead>
<tr>
<th></th>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Test 2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Test 3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Test 4</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Test 5</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Test 6</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Test 7</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Test 8</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Test 9</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Objective and Results

• Problem with Combinatorial Testing
  - Constructing a minimum test set is computationally infeasible (NP-Complete).

• Our Results
  - Developing new heuristic algorithms based on artificial life techniques; namely,
    • Genetic Algorithm (GA)
    • Ant Colony Algorithm (ACA)
The AETG Algorithm

- An algorithm used in AETG, a commercial testing tool of Telcordia (http://aetgweb.argreenhouse.com/).
- Our algorithms are modifications of this algorithm.

\[
\text{TestSet := Empty;}
\]
\[
\text{repeat \{ Construct a new test. Add the test to TestSet; \}}
\]
\[
\text{until (all combinations are covered by TestSet;)}
\]
Sub-Problem Tackled

- **Input**
  - A test set.

- **Output**
  - A single test

- **Objective**
  - To cover as many new combinations as possible

- **AETG**
  - A random greedy algorithm

- **Ours**
  - Genetic Algorithm (GA)
  - Ant Colony Algorithm (ACA)
Genetic Algorithm (GA)

• A metaheuristic algorithm that mimics the natural selection process
  - Solution = Chromosome

Selection
Selects good solutions

Crossover or Mutation
Creates a new generation
Proposed Genetic Algorithm (GA)

- **Chromosome**
  - A Test Case

- **Crossover operator**
  - Exchanges the values for each parameter with probability 0.5 (Uniform Crossover)

- **Mutation Operator**
  - Randomly changes the value of each parameter
Ant Colony Algorithm (ACO)

- A metaheuristic algorithm that mimics the pheromone-trail building of an ant swarm

- Ants randomly travel in the graph representing the solution space.
  - A path followed by an ant represents a solution.
  - An arc with a high amount of pheromone is chosen with high probability.

- Pheromone is deposited on the path followed.
  - The amount of pheromone deposited depends on the quality of the corresponding solution.

Ants will eventually converge to a path representing a good solution.
Proposed Ant Colony Algorithm (ACA)

• **Graph representation of the solution space**
  - Each path from the leftmost node to the rightmost node represents a test case.

• **The amount of pheromone deposited**
  - Proportional to the number of the newly covered combinations
Experiments

• Algorithms compared
  – The original AETG algorithm
  – The two proposed algorithm
    • Genetic Algorithm (GA)
    • Ant Colony Algorithm (ACA)

• Termination Condition
  – No more than 1000 candidate solutions were evaluated in creating a single test.

• Problem Instances
  – 16 instances were taken from the literature.
  – 300 runs were executed for each instance.
## Results (Pairwise Combinations, \( k=2 \))

### Number of Tests Required to Cover All Combinations

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Tests Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 ^4</td>
<td>11</td>
</tr>
<tr>
<td>3 ^13</td>
<td>17</td>
</tr>
<tr>
<td>2 ^100</td>
<td>*</td>
</tr>
<tr>
<td>10 ^20</td>
<td>*</td>
</tr>
<tr>
<td>5 ^1 * 3 ^8 * 2 ^2</td>
<td>*</td>
</tr>
<tr>
<td>7 ^1 * 6 ^1 * 5 ^1 * 4 ^5 * 3 ^8 * 2 ^3</td>
<td>*</td>
</tr>
<tr>
<td>5 ^1 * 4 ^1 * 3 ^11 * 2 ^5</td>
<td>*</td>
</tr>
<tr>
<td>6 ^1 * 5 ^1 * 4 ^1 * 6 ^3 * 8 ^2 ^3</td>
<td>*</td>
</tr>
<tr>
<td>4 ^1 * 15 ^1 * 17 ^1 * 2 ^29</td>
<td>*</td>
</tr>
<tr>
<td>4 ^1 * 3 ^39 * 2 ^35</td>
<td>*</td>
</tr>
</tbody>
</table>

*The figures are taken from Cohen et al. “New techniques for designing qualitatively independent systems,” ICSE 2003.

1 5-valued parameter
8 3-valued parameters
2 2-valued parameters

1212
## Results (3-Way Combinations, \( k=3 \))

### Number of Tests Required to Cover All Combinations

<table>
<thead>
<tr>
<th></th>
<th>221</th>
<th>1490</th>
<th>335</th>
<th>64</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>47</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>6</td>
<td>47</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>10</td>
<td>105</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
<tr>
<td>5</td>
<td>105</td>
<td>⬤</td>
<td>⬤</td>
<td>⬤</td>
</tr>
</tbody>
</table>

*The figures are taken from Cohen et al. “New techniques for designing qualitatively independent systems,” ICSE 2003.*
Conclusions

- Two algorithms were proposed for combinatorial testing.
  - Genetic Algorithm
  - Ant Colony Algorithm

- Experiments were conducted.
  - For the case $k = 2$ (pairwise combinations), the two algorithms exhibited comparable performance to the AETG algorithm.
  - For the case $k = 3$ (3-way combinations), the two algorithms outperformed the AETG algorithm.