Comparing Metaheuristic Algorithms for Error Detection in Java Programs

Francisco Chicano, Marco Ferreira and Enrique Alba
Motivation

• Concurrent software is difficult to test ...
• ... and it is in the heart of a lot of critical systems

• Techniques for proving the correctness of concurrent software are required
• Model checking → fully automatic
• Traditional techniques for this purpose have problems with large models
• We compare several metaheuristics and classical algorithms for model checking
• **Objective**: Prove that model $M$ satisfies the property $f : M \models f$

• In the general case, $f$ is a temporal logic formula (LTL, CTL, etc.)
Safety properties

\[ \forall \sigma \in S^\omega : \sigma \not\in \mathcal{P} \Rightarrow (\exists i \geq 0 : \forall \beta \in S^\omega : \sigma_i \beta \not\in \mathcal{P}) \]

- An error trail is an execution path ending in an error state
- The search for errors is transformed in a graph exploration problem (DFS, BFS)

Properties in JPF
- Exceptions
- Deadlocks
State Explosion Problem

- **Number of states** very large even for small models

- **Example:** Dining philosophers with \( n \) philosophers \( \rightarrow 3^n \) states

- For each state we need to store the heap and the stacks of the different threads

- **Solutions:** collapse compression, minimized automaton representation, bitstate hashing, partial order reduction, symmetry reduction

- Large models cannot be verified but errors can be found
The search for errors can be directed by using heuristic information.

Different kinds of heuristic functions have been proposed in the past:

- Formula-based heuristics
- Structural heuristics
- Deadlock-detection heuristics
- State-dependent heuristics
Classification of Algorithms

- **Guided**
  - Complete: BFS, DFS
  - Non-complete: A*, RDFS, RS

- **Non-guided**
  - Complete: EDA
  - Non-complete: GA, GAMO, PSO, SA, ACO

- **Deterministic**
  - BFS, DFS

- **Stochastic**
  - EDA

(working on)
Genetic Algorithm

\[ P = \text{generateInitialPopulation}(); \]
\[ \text{evaluate}(P); \]
\[ \text{while} \ \text{not stoppingCondition}() \ \text{do} \]
\[ P' = \text{selectParents}(P); \]
\[ P' = \text{applyVariationOperators}(P'); \]
\[ \text{evaluate}(P'); \]
\[ P = \text{selectNewPopulation}(P, P'); \]
\[ \text{end while} \]
\[ \text{return} \ \text{the best found solution} \]

**Solution encoding**
(floating point values)

\[ 0.5 \ 0.1 \ 0.9 \ 0.3 \ 0.5 \ 0.9 \]

**Crossover**

\[ 0.5 \ 0.1 | 0.9 \ 0.3 \ 0.5 \ 0.9 \]
\[ 0.2 \ 0.6 \ 0.1 \ 0.7 \ 0.8 \ 0.4 | 0.2 \ 0.0 \ 0.6 \]

**Mutation**

\[ 0.5 \ 0.1 \ 0.9 \ 0.3 \ 0.5 \ 0.9 \ \rightarrow \ 0.5 \ 0.1 \ 0.6 \ 0.3 \ 0.5 \ 0.9 \]
Genetic Algorithm with Memory Operator

Solution encoding
(floating point values)

0.5 0.1 0.9 0.3 0.5 0.9

Index in a table of states
Particle Swarm Optimization

\[ P = \text{generateInitialPopulation}(); \]
\[ \text{while} \ \text{not stoppingCondition}(P) \ \text{do} \]
\[ \quad \text{evaluate}(P); \]
\[ \quad \text{calculateNewVelocityVectors}(P); \]
\[ \quad \text{move}(P); \]
\[ \text{end while} \]
\[ \text{return} \ \text{the best found solution} \]

Particles

\[
\begin{align*}
0.2 & \rightarrow \text{Position (solution)} \\
1.0 & \rightarrow \text{Velocity}
\end{align*}
\]

Inertia

\[
\begin{align*}
\dot{v}_{ij}(t + 1) &= w \cdot \dot{v}_{ij}(t) + c_1 \cdot r_1 \cdot (p_{ij} - \dot{x}_{ij}(t)) + c_2 \cdot r_2 \cdot (n_{ij} - \dot{x}_{ij}(t)) \\
\dot{x}_{ij}(t + 1) &= \dot{x}_{ij}(t) + \dot{v}_{ij}(t + 1)
\end{align*}
\]

Global best

Personal best
Ant Colony Optimization

procedure ACOMetaheuristic
  ScheduleActivities
  ConstructAntsSolutions
  UpdatePheromones
  DaemonActions // optional
end ScheduleActivities
end procedure

- The ant selects stochastically its next node
- The probability of selecting one node depends on the pheromone trail and the heuristic value (optional) of the edge/node
- The ant stops when a complete solution is built

\[
p_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{x \in N_i}[\tau_{ix}]^\alpha [\eta_{ix}]^\beta}
\]
Simulated Annealing

\[
S = \text{generateInitialSolution}();
T = \text{initialTemperature};
\text{while not stoppingCondition()} \text{ do}
\quad N = \text{getRandomNeighbor}(S);
\quad \Delta E = \text{energy}(N) - \text{energy}(S);
\quad \text{if } \Delta E > 0 \text{ OR } \text{random}(0,1) < \text{probabilityAcceptance}(\Delta E, T) \text{ then}
\quad \quad S = N
\quad \text{end if}
\quad T = \text{updateTemperature}(T);
\text{end while}
\text{return } S
\]

\[
\text{probabilityAcceptance}(\Delta E, T) = e^{\frac{\Delta E}{T}}
\]

Neighbor

0.5 0.1 0.9 0.3 0.5 0.9 \rightarrow 0.5 0.1 0.6 0.3 0.5 0.9
Parameterization

- We used 3 scalable and 2 non-scalable models for the experiments

<table>
<thead>
<tr>
<th>Program</th>
<th>Loc</th>
<th>Functions</th>
<th>Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>dinj</td>
<td>63</td>
<td>j=4 to 20</td>
<td>j+1</td>
</tr>
<tr>
<td>phi j</td>
<td>176</td>
<td>j=4 to 36</td>
<td>j+1</td>
</tr>
<tr>
<td>mar j</td>
<td>186</td>
<td>j=3</td>
<td>j+1</td>
</tr>
<tr>
<td>giop</td>
<td>74</td>
<td>j=2 to 10</td>
<td>7</td>
</tr>
<tr>
<td>gar p</td>
<td>458</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

- Maximum number of expanded states: 200 000
- Fitness function:

\[ f(x) = \text{deadlock} + \text{numblocked} + \frac{1}{1 + \text{pathlen}} \]

- 100 independent executions of stochastic algorithms
## Hit Rate

<table>
<thead>
<tr>
<th>Problem</th>
<th>DFS</th>
<th>BFS</th>
<th>$A^*$</th>
<th>GA</th>
<th>GAMO</th>
<th>PSO</th>
<th>SA</th>
<th>ACO_{hg}</th>
<th>RS</th>
<th>BS</th>
</tr>
</thead>
<tbody>
<tr>
<td>phi 4</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>phi 12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>phi 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>phi 28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>phi 36</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>82</td>
<td>100</td>
<td>53</td>
<td>79</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>din 4</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>din 8</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>76</td>
<td>100</td>
<td>96</td>
<td>100</td>
</tr>
<tr>
<td>din 12</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>96</td>
<td>85</td>
<td>13</td>
<td>68</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>din 16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91</td>
<td>58</td>
<td>20</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>din 20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>52</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>mar 2</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>mar 4</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>96</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>mar 6</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>mar 8</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>mar 10</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>25</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>giop</td>
<td>100</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>68</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>garp</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>2</td>
<td>80</td>
<td>87</td>
<td>87</td>
<td>100</td>
<td>0</td>
</tr>
</tbody>
</table>
Hit rate

Parameterization  Hit Rate  Length of Error Trails

SSBSE 2011, Szeged, Hungary, September 10-12
Length of Error Trails

BS: 753
Length of Error Trails

- BS: 172
- BS: 260
- DFS: 245
- DFS: 378
Length of Error Trails

The graph shows the length of error trails for different algorithms. The x-axis represents the algorithms: DFS, BFS, A*, GA, GAMO, PSO, SA, ACO, RS, BS, and GERP. The y-axis represents the hit rate, with values ranging from 0 to 400. The graph compares the performance of these algorithms in terms of the length of error trails.
Conclusions

- Metaheuristics are more effective than classical algorithms in finding errors
- Beam Search has advantages over complete search algorithms
- An even distribution of the search in depth levels tends to raise hit rate
- Stochastic algorithms obtain short error trails

Future Work

- Design a stochastic complete guided algorithm to find errors and verify
- Design of hybrid algorithms to more efficiently explore the search space
- Explore the design of parallel metaheuristics for this problem
Thanks for your attention !!!

Comparing Metaheuristic Algorithms for Error Detection in Java Programs