Table of Contents

1 Introduction
2 The Test Case Generator
3 Distance for instanceof
4 Experiments
5 Conclusions
After codification, software products require a test phase

The objective is to find errors

Object Oriented paradigm is followed by most software developers

Inheritance is an important issue of this paradigm

We propose a distance measure for the `instanceof` operator that use the information of the class hierarchy

We define two mutation operators based on the distance
The test case generator breaks the global objective into several partial objectives

Our generator creates a coverage table with the values that traverse each branch

```java
Genetic Algorithm

t := 0;
P(t) = Generate ();
Evaluate (P(t));
while not StopCriterion do
    P'(t) := VariationOps (P(t));
    Evaluate (P'(t));
    P(t+1) := Replace (P'(t),P(t));
    t := t+1;
endwhile;
```
We defined an **objective function** (fitness) to be minimized

\[ f_b(x) = \begin{cases} bd_b(x) & \text{if } b \text{ is traversed by } x \\ bd_c(x) + ap(c, b) + pen & \text{otherwise} \end{cases} \]

\[ bd(a \& b) = bd(a) + bd(b) \]

\[ bd(a | b) = \min(bd(a), bd(b)) \]
Distance for `instanceof`

Hierarchical walk = 3
Approximation walk = 2

\[ d = 3h + 2a \]

\[ d(c, r) = h|w_{c' \rightarrow c}| + a|w_{c' \rightarrow r}|, \text{ if } r \in C_R \]

**ArrayList** `instanceof** Integer**
Distance for instanceof

\[ d(c, r) = h|w_{c' \rightarrow c}| + a|w_{c' \rightarrow r}|, \text{ if } r \in C_R \]

**ArrayList** instanceof **Integer**

**ArrayList** instanceof **Set**
Distance-based and Uniform Mutation

**Uniform Mutation**

\[
p(c, c') = \begin{cases} 
\frac{1}{|U|-1} & \text{if } c \neq c' \\
0 & \text{if } c = c'
\end{cases}
\]

**Distance-based Mutation**

\[
p(c, c') = \begin{cases} 
\frac{1}{\sum_{r \in U, r \neq c} \frac{d(c, r)}{d(c, c')}} & \text{if } c \neq c' \\
0 & \text{if } c = c'
\end{cases}
\]
Experiments: Programs

Test Programs instanceof

instanceof expression

Named: Obj i_j
Atomic Conditions (i): 2-4
Nesting degree (j): 1-3
Experiments: Approximation and Hierarchical Constants

**Table of Coverage**

<table>
<thead>
<tr>
<th></th>
<th>$h = 1$</th>
<th>$h = 25$</th>
<th>$h = 50$</th>
<th>$h = 100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a = 200$</td>
<td>75.45 %</td>
<td>75.33 %</td>
<td>74.93 %</td>
<td>75.68 %</td>
</tr>
<tr>
<td>$a = 100$</td>
<td>75.53 %</td>
<td>74.74 %</td>
<td>75.10 %</td>
<td>74.79 %</td>
</tr>
<tr>
<td>$a = 50$</td>
<td>74.85 %</td>
<td>75.81 %</td>
<td>74.44 %</td>
<td>73.56 %</td>
</tr>
</tbody>
</table>

Does it hold that $a > h$?

Yes, because $a$ weights how close the test case is to satisfy the condition.
Experiments: Distance-based and Uniform mutation

Fitness evolution with a random uniform initialization

- Average of 200 executions
Experiments: Distance-based and Uniform Mutation

Fitness evolution with greedy seeding initialization

- Average of 200 executions
- MDn is better than MU
- MU is faster at the beginning

New proposal: 

Adaptive Mutation
Adaptive Mutation: New Proposal

\[
p(c, c') = \begin{cases} 
\frac{1}{d(c,c')} & \text{if } c \neq c' \\
\sum_{r \in U, r \neq c} \frac{1}{d(c,r)} & \text{if } c = c' \\
0 & \text{if } c = c'
\end{cases}
\]

Mutation probability VS Distance between classes

\(\alpha = 0\)

\(\alpha = 1\)

\(\alpha = 2\)

Adaptive Speed

\[\alpha = \lambda \cdot \text{step}\]
Adaptive Mutation: Experiments

- 100% coverage obtained in all programs
- The GA with adaptive mutation is much better than the Random Search
- Difficult to test is correlated to the expression $i + j$
Conclusions & Future Work

- We created a test case generator able of dealing with inheritance
- A new branch distance has been defined for inheritance
- We have proposed and compared two mutation operators based on the distance
- The MDn operator is better when using a greedy seeding of the GA
- The number of atomic conditions and the nesting degree have a great influence on the automatic testing complexity

- Combine our proposal with other OO features
- Analysis of the impact of our proposal in real-world software
THANKS FOR YOUR ATTENTION

Javier Ferrer
ferrer@lcc.uma.es