# Dealing with Inheritance in OO Evolutionary Testing

Javier Ferrer , Francisco Chicano and Enrique Alba

ferrer@lcc.uma.es chicano@lcc.uma.es eat@lcc.uma.es

UNIVERSIDAD DE MÁLAGA

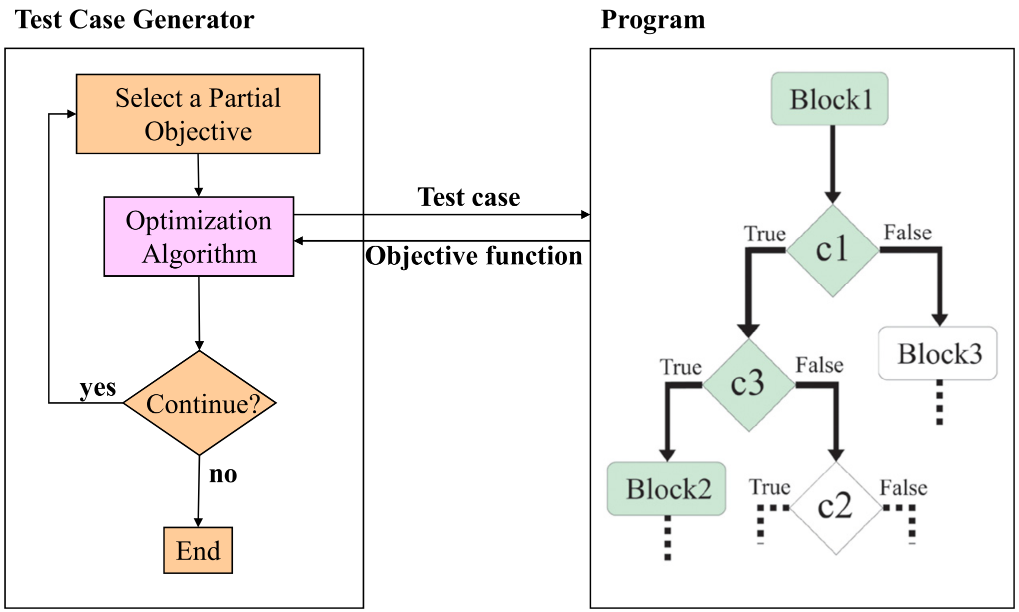
# Table of Contents

1. Introduction
2. The Test Case Generator
3. Distance for `instanceof`
4. Experiments
5. Conclusions

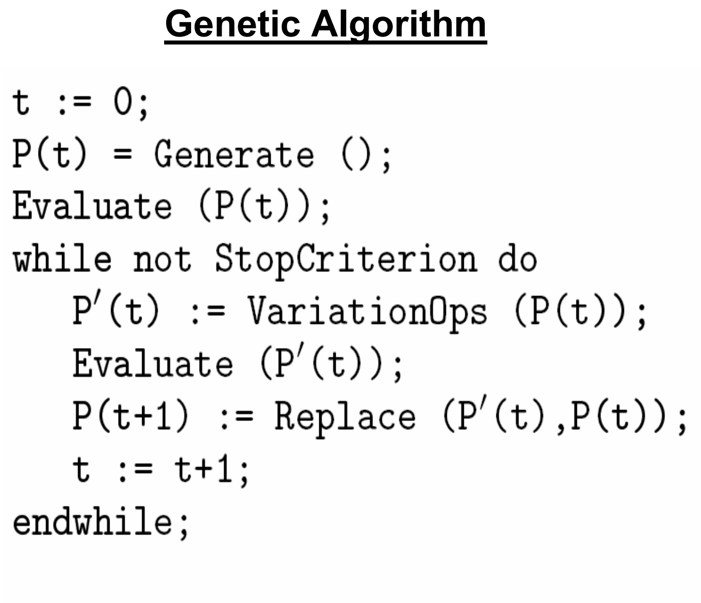
# Introduction

- 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



## 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;
```

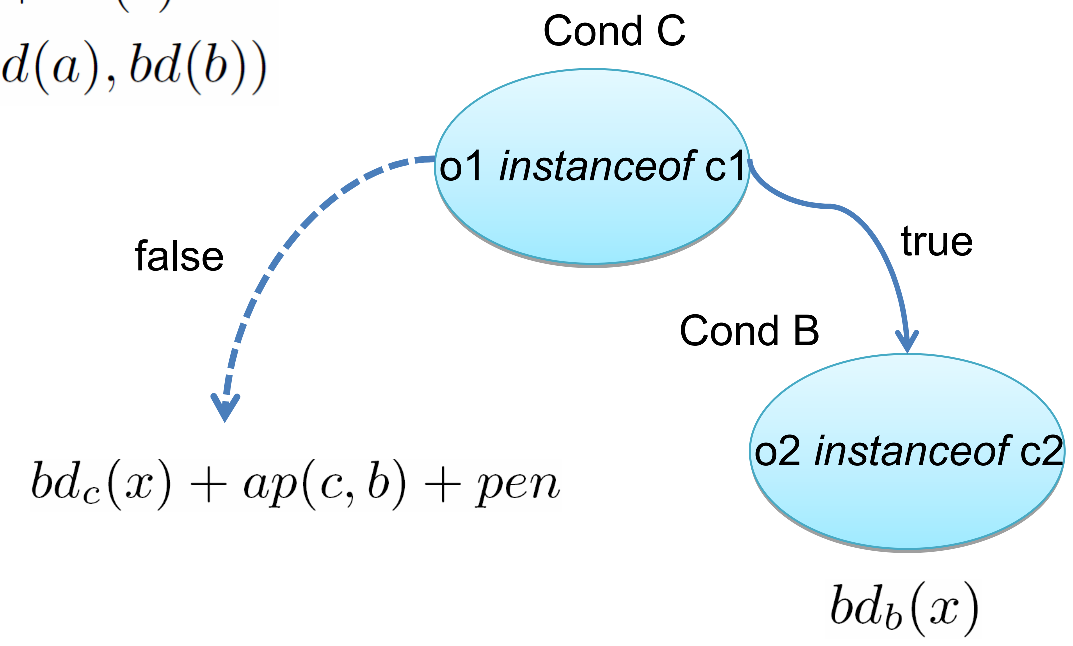
- The test case generator breaks the global objective into several partial objectives
- Our generator creates a coverage table with the values that traverses each branch

# Distance for `instanceof`

- We defined an ***objective function*** (fitness) to be minimized

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

$$\begin{aligned} bd(a\&b) &= bd(a) + bd(b) \\ bd(a \,|\, b) &= \min(bd(a), bd(b)) \end{aligned}$$



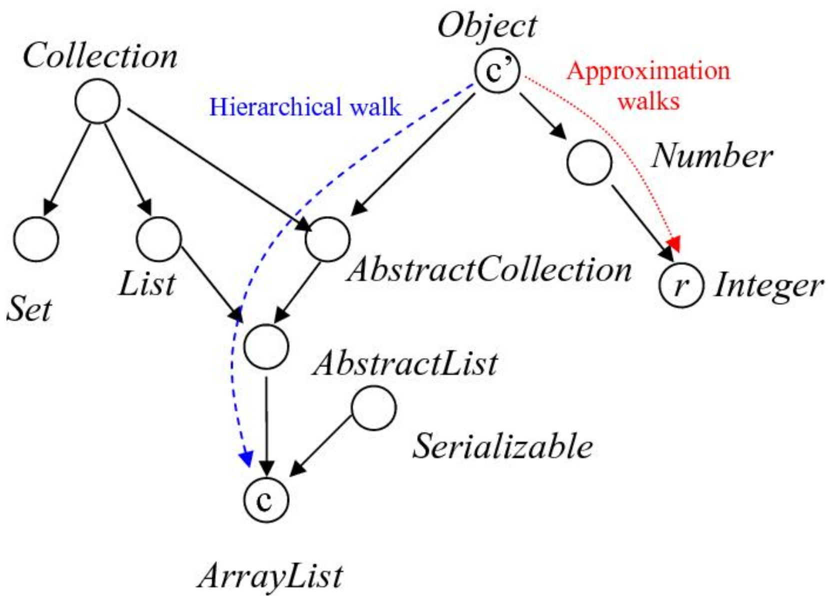
# Distance for instanceof

C instanceof R

CLASS

Hierarchical walk= 3
Approximation walk= 2

d=3h+2a



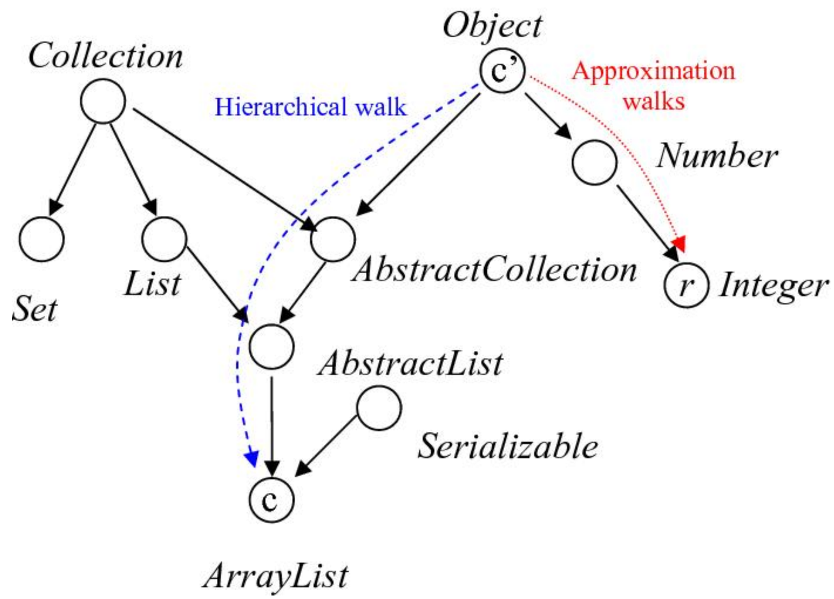
$$\boxed{d(c,r) = h|w_{c' \rightarrow c}| + a|w_{c' \rightarrow r}| \text{ , if } r \in C_R}$$

**ArrayList** instanceof **Integer**

# Distance for instanceof

## C instanceof R

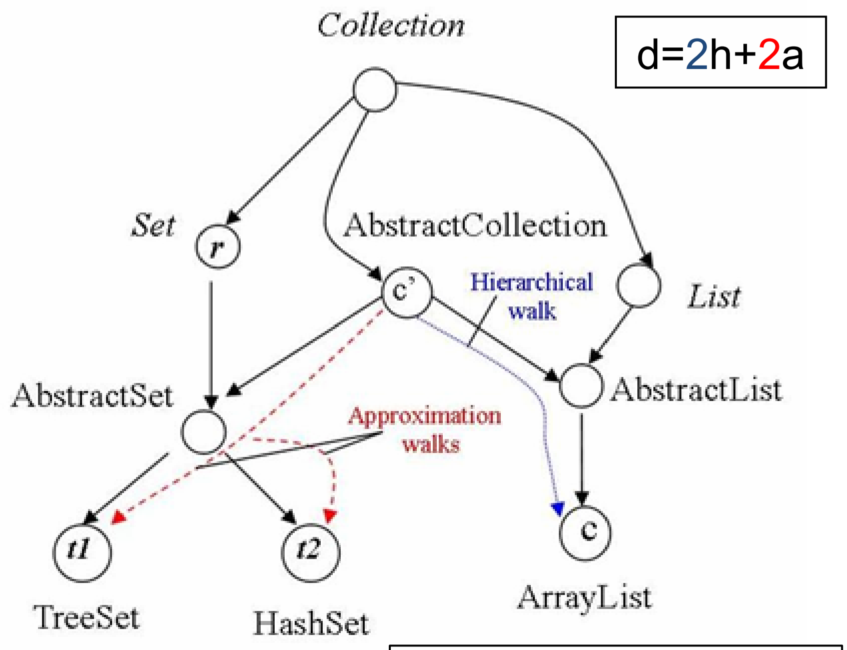
### CLASS



$$d(c,r) = h|w_{c'\to c}| + a|w_{c'\to r}| \quad \text{, if } r \in C_R$$

**ArrayList** instanceof **Integer**

### INTERFACE

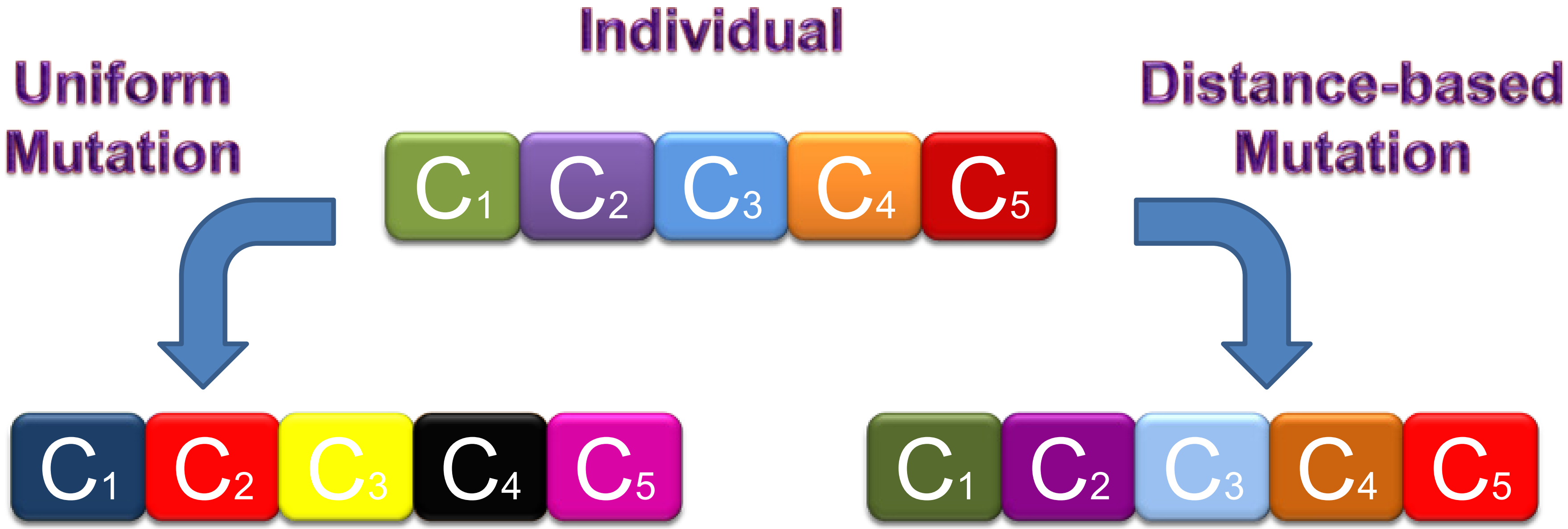


d=2h+2a

$$d(c,r) = \min_{t \in S_r} d(c,t)$$

**ArrayList** instanceof **Set**

# Distance-based and Uniform Mutation

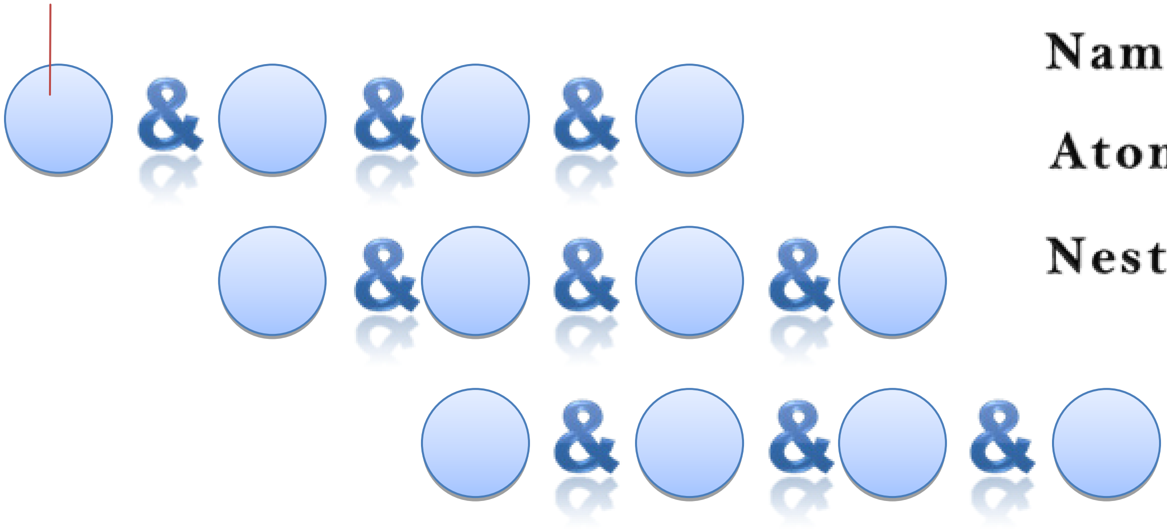


$$p(c,c') = \begin{cases} \frac{1}{|U|-1} & \text{if } c \neq c' \\ 0 & \text{if } c = c' \end{cases}$$

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

# Experiments: Programs

## Test Programs instanceof



**intanceof** expression

Named: Obj i_ j

Atomic Conditions (i): 2-4

Nesting degree (j): 1-3

# Experiments : Approximation and Hierarchical Constants

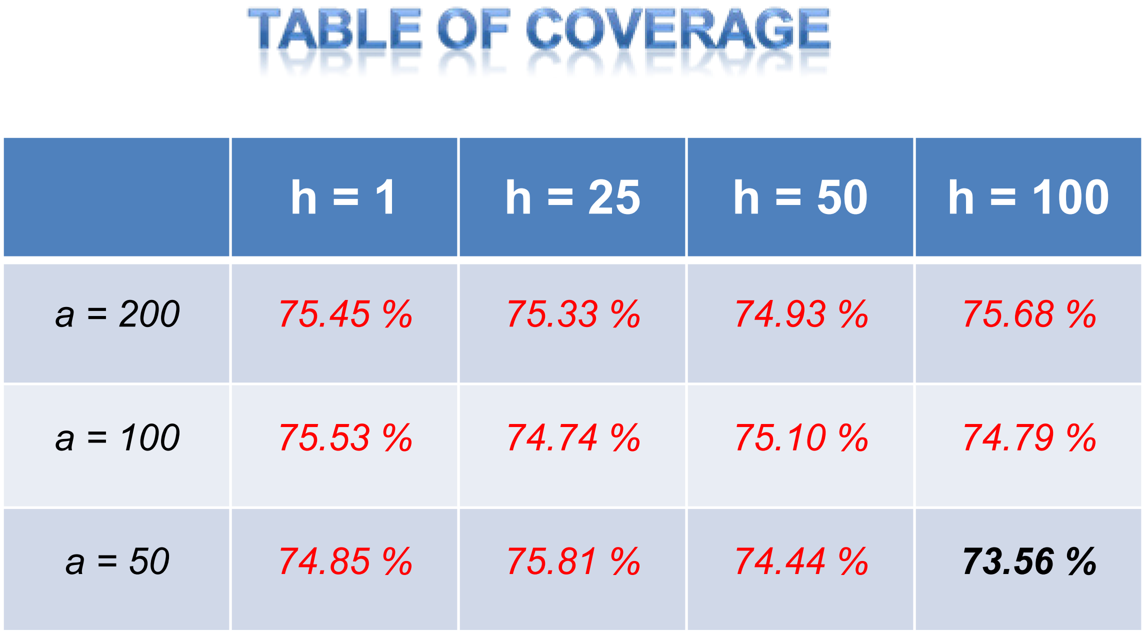


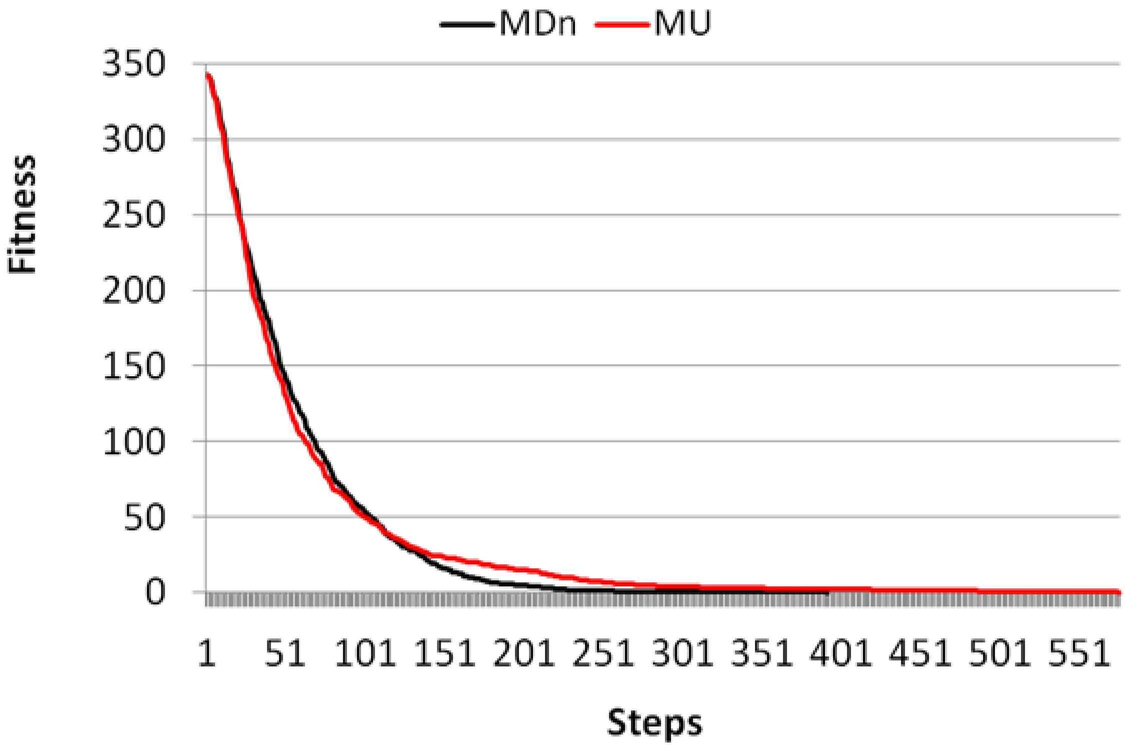
TABLE OF COVERAGE

| | h = 1 | h = 25 | h = 50 | h = 100 |
|---|---|---|---|---|
| *a = 200* | *75.45 %* | *75.33 %* | *74.93 %* | *75.68 %* |
| *a = 100* | *75.53 %* | *74.74 %* | *75.10 %* | *74.79 %* |
| *a = 50* | *74.85 %* | *75.81 %* | *74.44 %* | ***73.56 %*** |

- 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

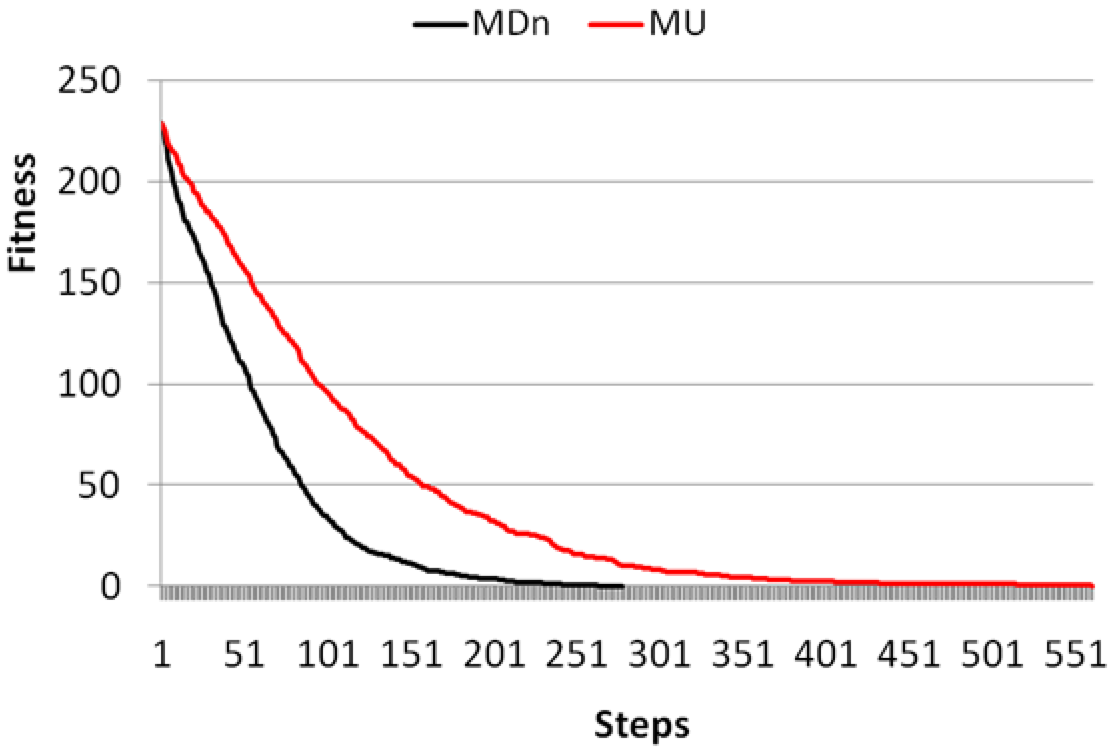
## Fitness evolution with a random uniform initialization



- 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**

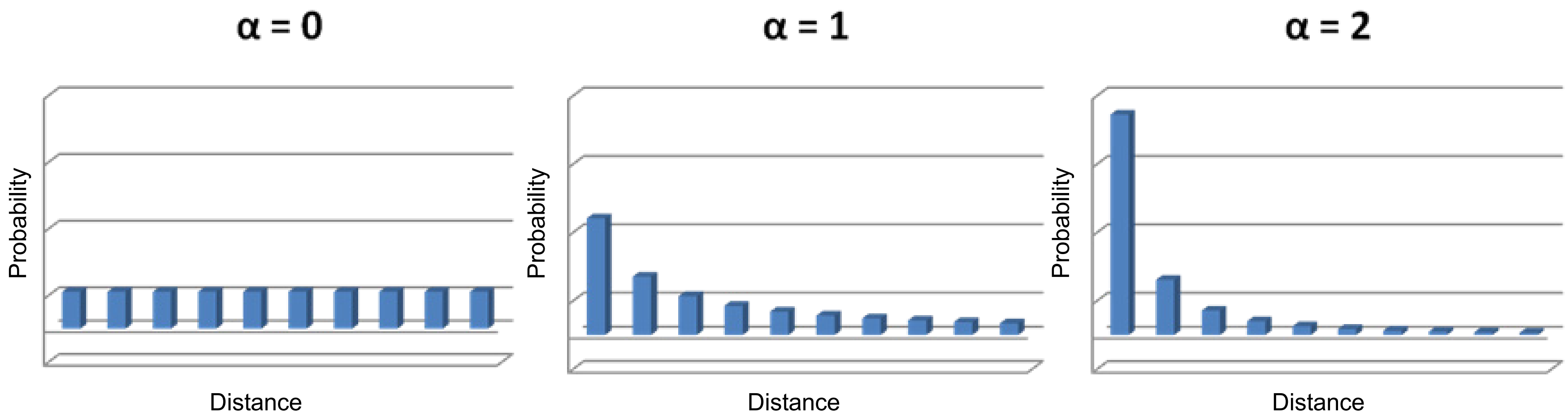
# Adaptive Mutation : New Proposal

$$p(c,c') = \begin{cases} \dfrac{\left(\frac{1}{d(c,c')}\right)^{\alpha}}{\sum_{r\in U, r\neq c}\left(\frac{1}{d(c,r)}\right)^{\alpha}} & \text{if } c \neq c' \\ 0 & \text{if } c = c' \end{cases}$$

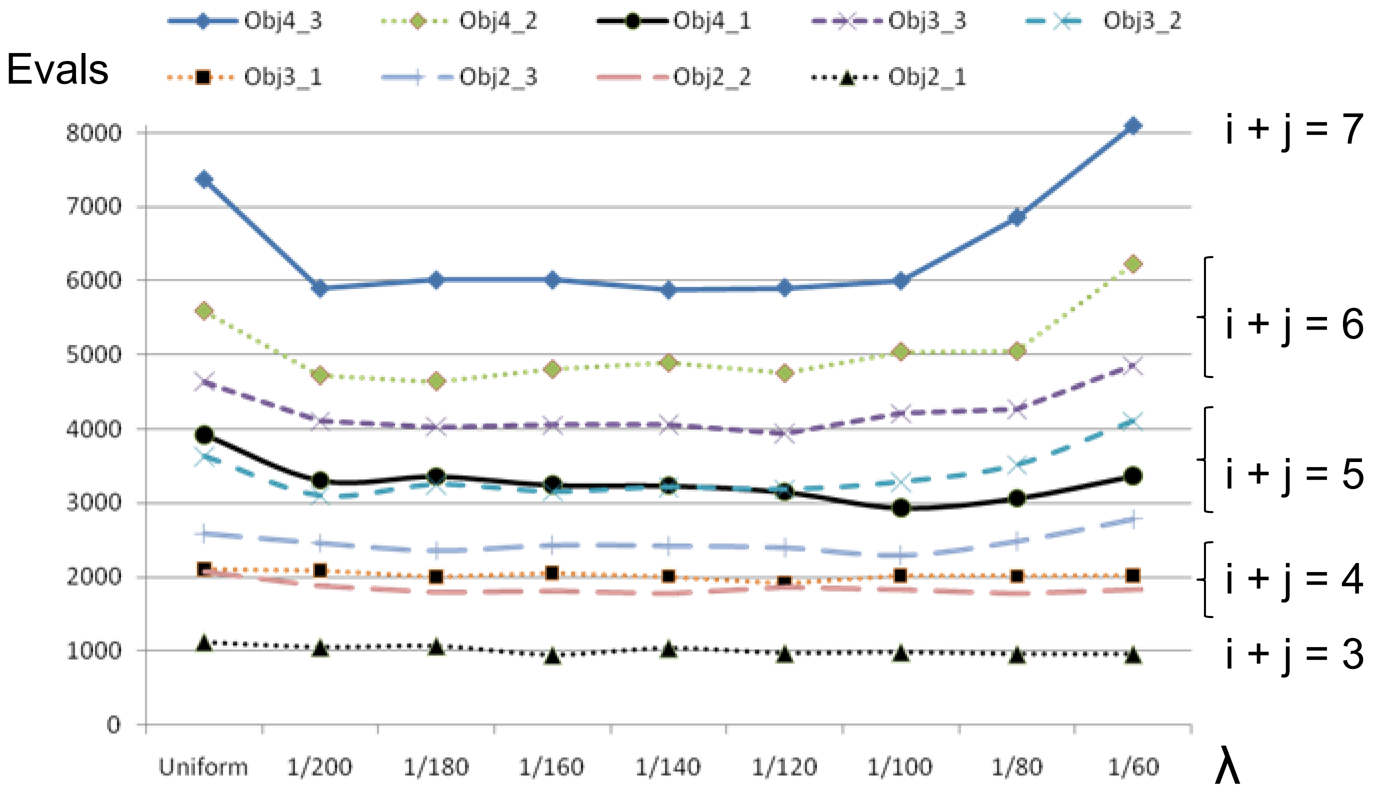
Adaptive Speed

$$\alpha = \lambda \cdot step$$

**Mutation probability VS Distance between classes**



# 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

# FINAL

THANKS FOR YOUR ATTENTION

Javier Ferrer
ferrer@lcc.uma.es