Evolutionary Testing of Autonomous Software Agents

Cu D. Nguyen, Anna Perini, and Paolo Tonella
Fondazione Bruno Kessler
Via Sommarive, 18 38050 Trento, Italy

Simon Miles, Mark Harman, and Michael Luck
Department of Computer Science
King’s College London
Strand, London WC2R 2LS, UK

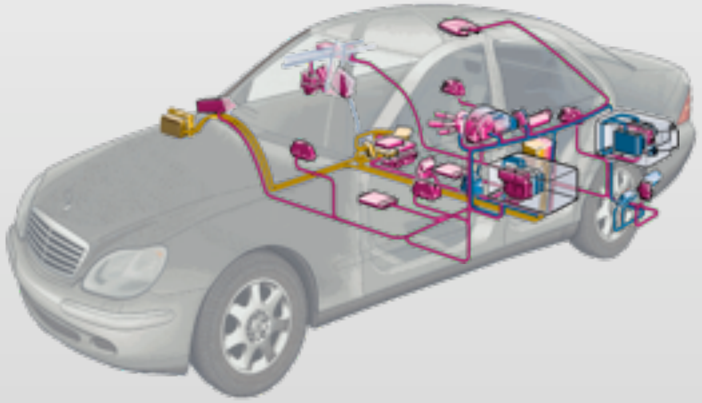
AAMAS, Budapest 2009

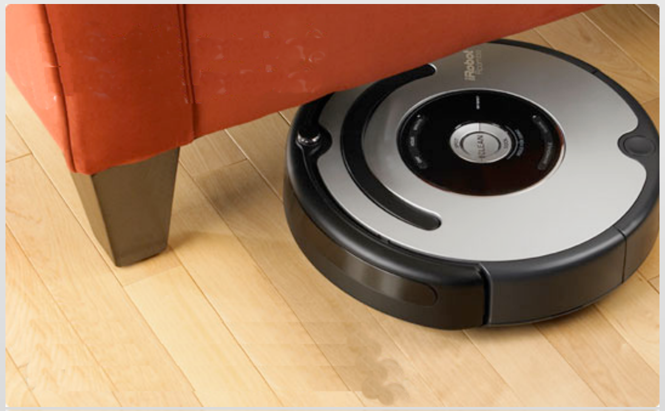
Outline

- Introduction
- Approaches
- Experiment and result discussion
- Conclusion

Introduction

Autonomous software agents are increasingly used





Testing to build confidence in their operations is crucial !

Introduction

Agent autonomy makes testing harder

- Agents make decisions for themselves based on their goals, intentions, and beliefs
- Can behave differently in response to the same input

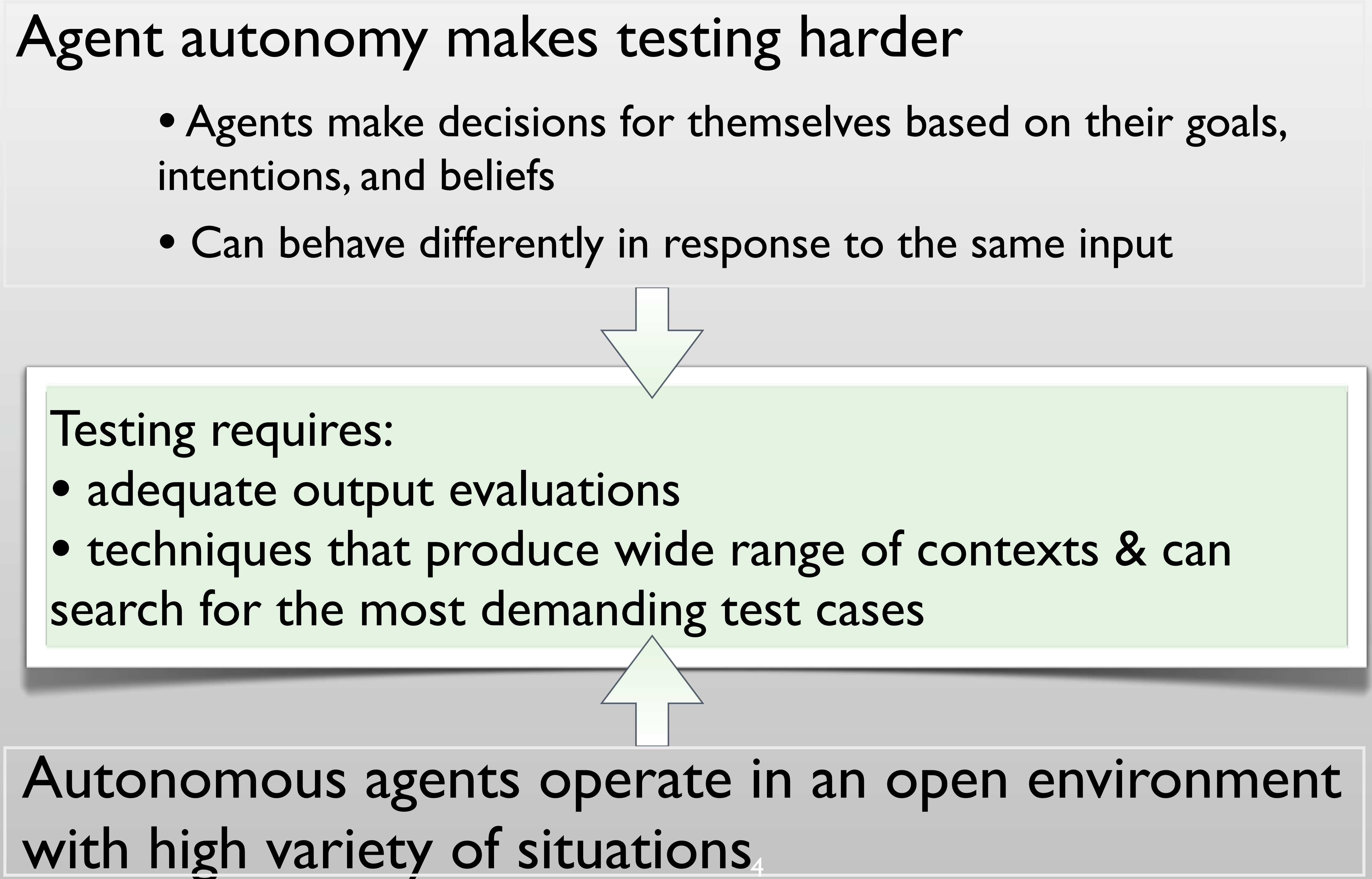
Introduction

Agent autonomy makes testing harder

- Agents make decisions for themselves based on their goals, intentions, and beliefs
- Can behave differently in response to the same input

Autonomous agents operate in an open environment with high variety of situations

Introduction



Agent autonomy makes testing harder

- Agents make decisions for themselves based on their goals, intentions, and beliefs
- Can behave differently in response to the same input

Testing requires:

- adequate output evaluations
- techniques that produce wide range of contexts & can search for the most demanding test cases

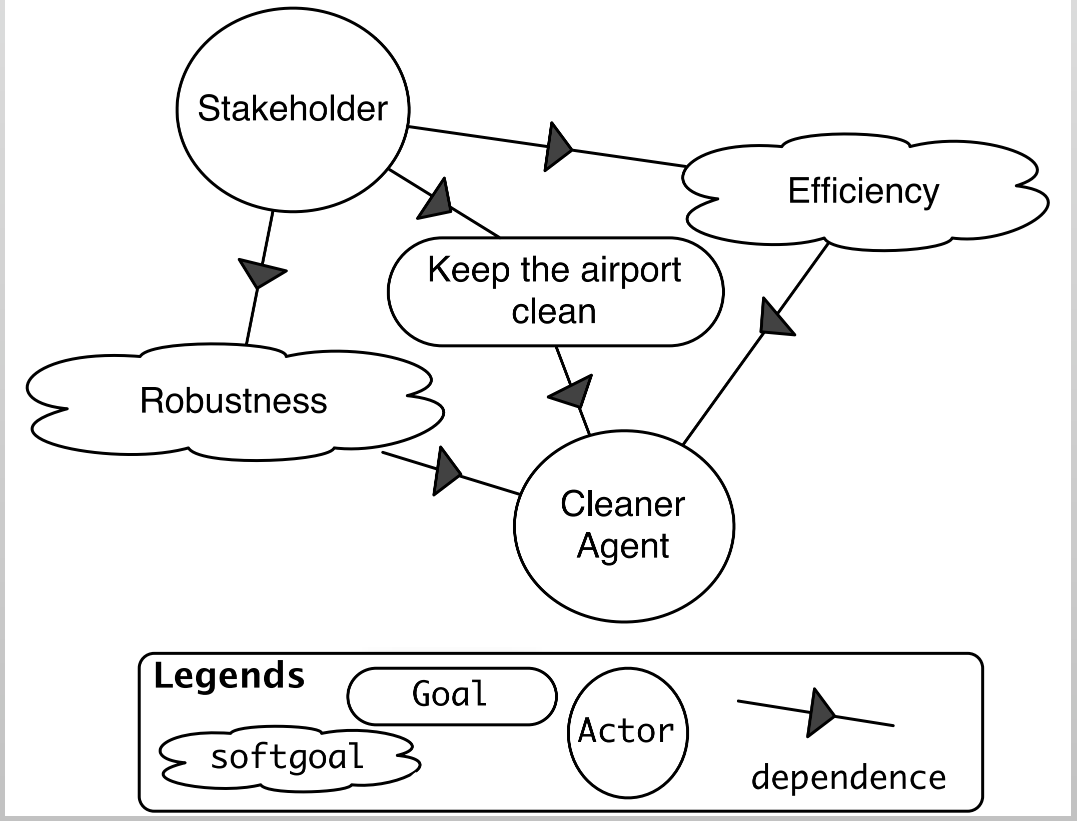
Autonomous agents operate in an open environment with high variety of situations

Background

- Testing is to find faults
- We focus on agent level
- We evaluate the exhibited performance of autonomous agents, not the underlying autonomy mechanism

Our approach (1)

Use stakeholder’ requirements related to quality (e.g. efficiency) to judge autonomous agents.



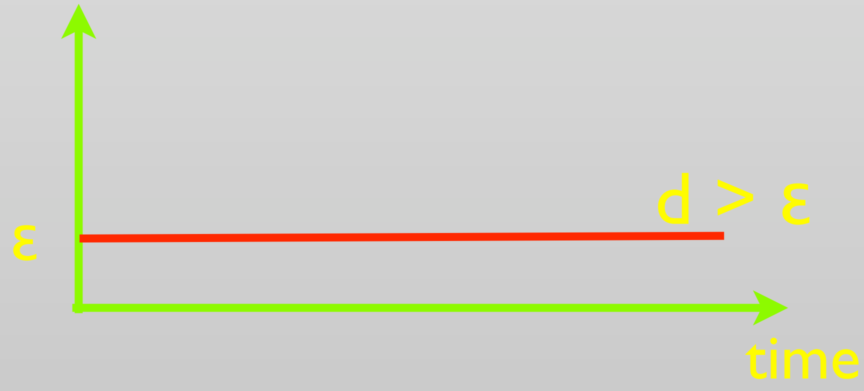
Our approach (1)

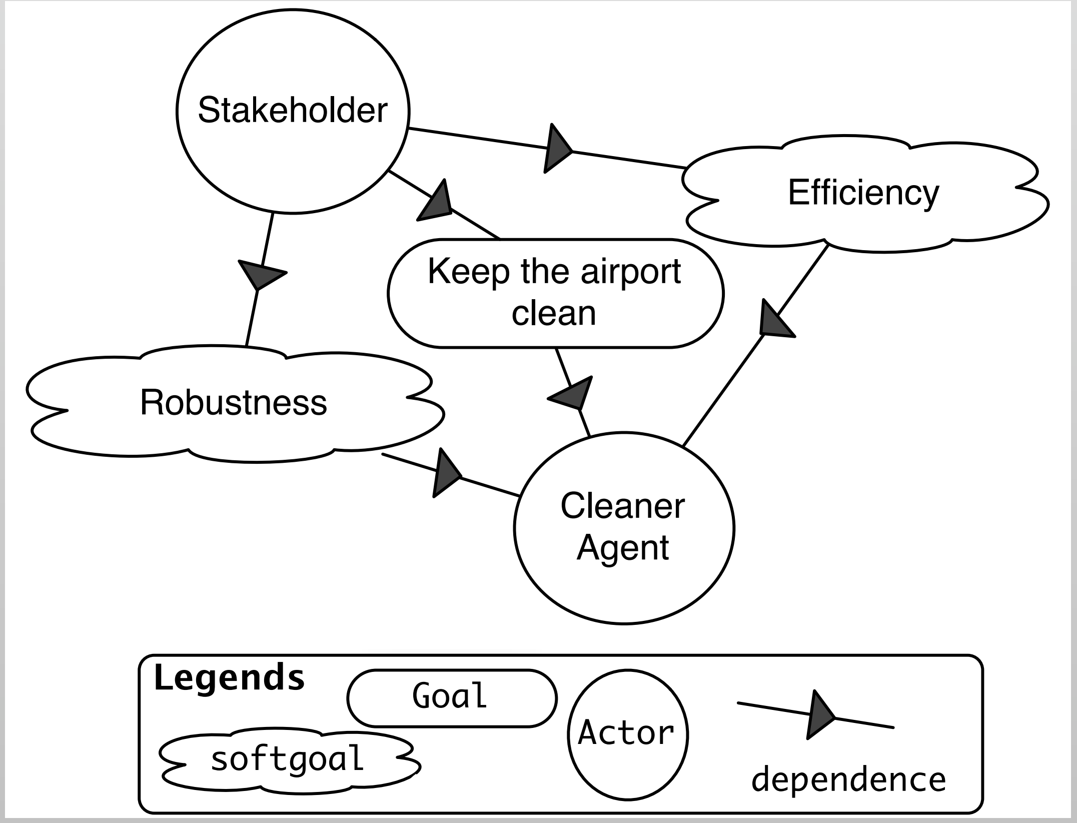
Use stakeholder' requirements related to quality (e.g. efficiency) to judge autonomous agents.

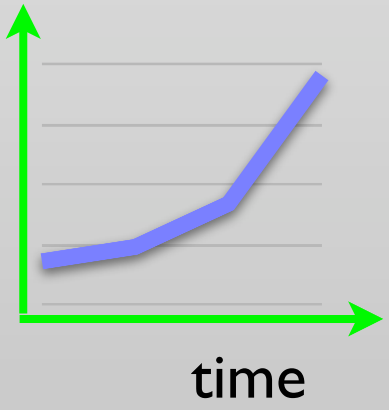
Represent these requirements as <u>quality functions</u>

Assess the agents under test

Drive the evolutionary generation







Our approach (2)

Use quality functions in fitness measures to drive the evolutionary generation

- Fitness of a test case tells how good the test case is
- Evolutionary testing searches for test cases having the best fitness values.

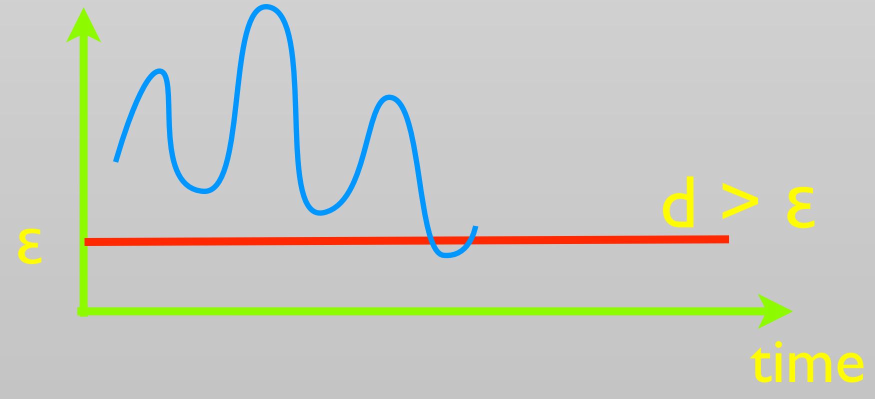
Fitness example: distance to be crashed

Our approach (2)

Use quality functions in fitness measures to drive the evolutionary generation

- Fitness of a test case tells how good the test case is
- Evolutionary testing searches for test cases having the best fitness values.

Fitness example: distance to be crashed

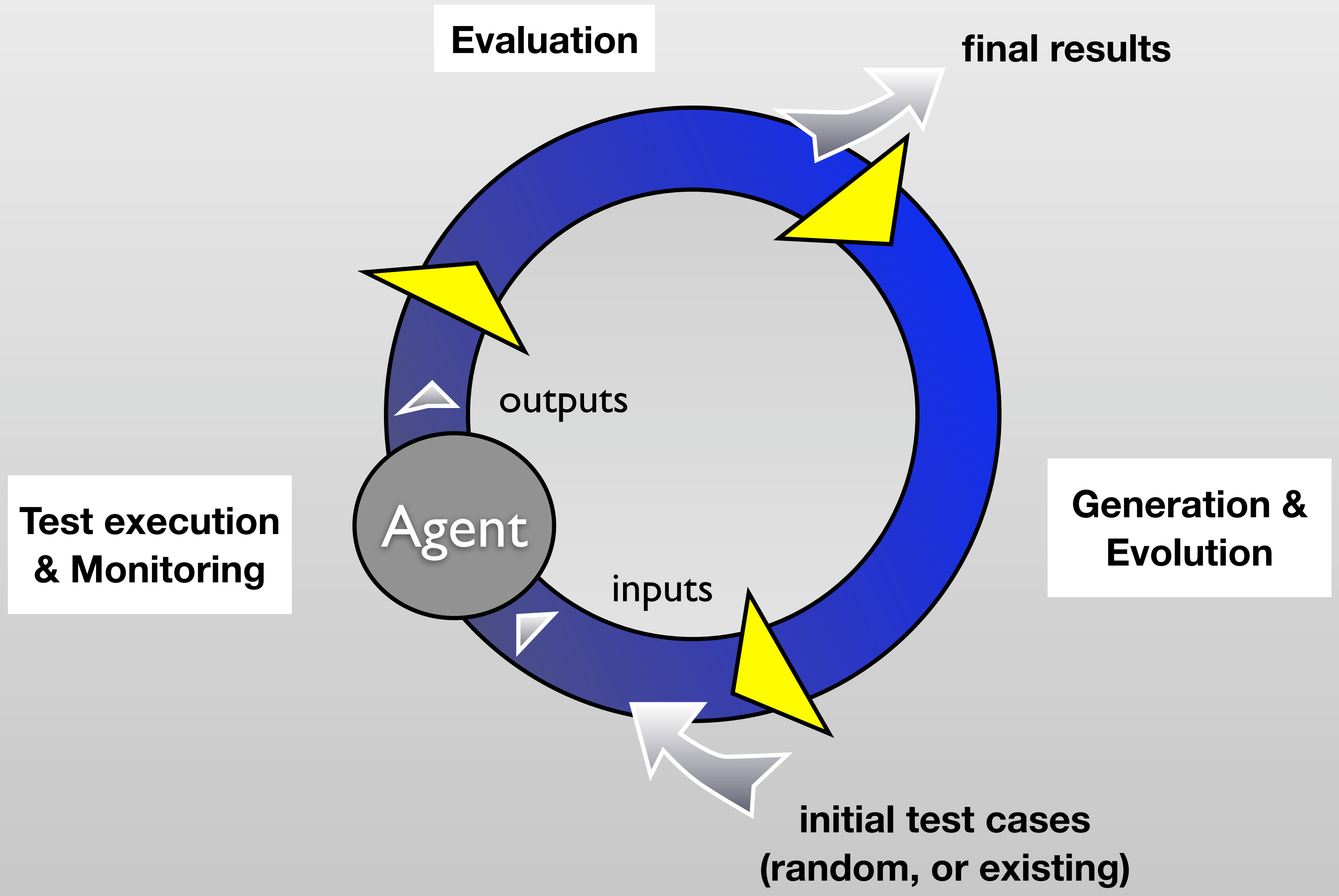


Our approach (3)

Use statistical methods to measure test case fitness

- Test outputs of a test case can be different
- A test case execution is repeated a number of times (or in parallel)
- Statistical output data are used to calculate the fitness

Evolutionary procedure



Evaluation

final results

outputs

Agent

Test execution & Monitoring

inputs

Generation & Evolution

initial test cases (random, or existing)

Experiments

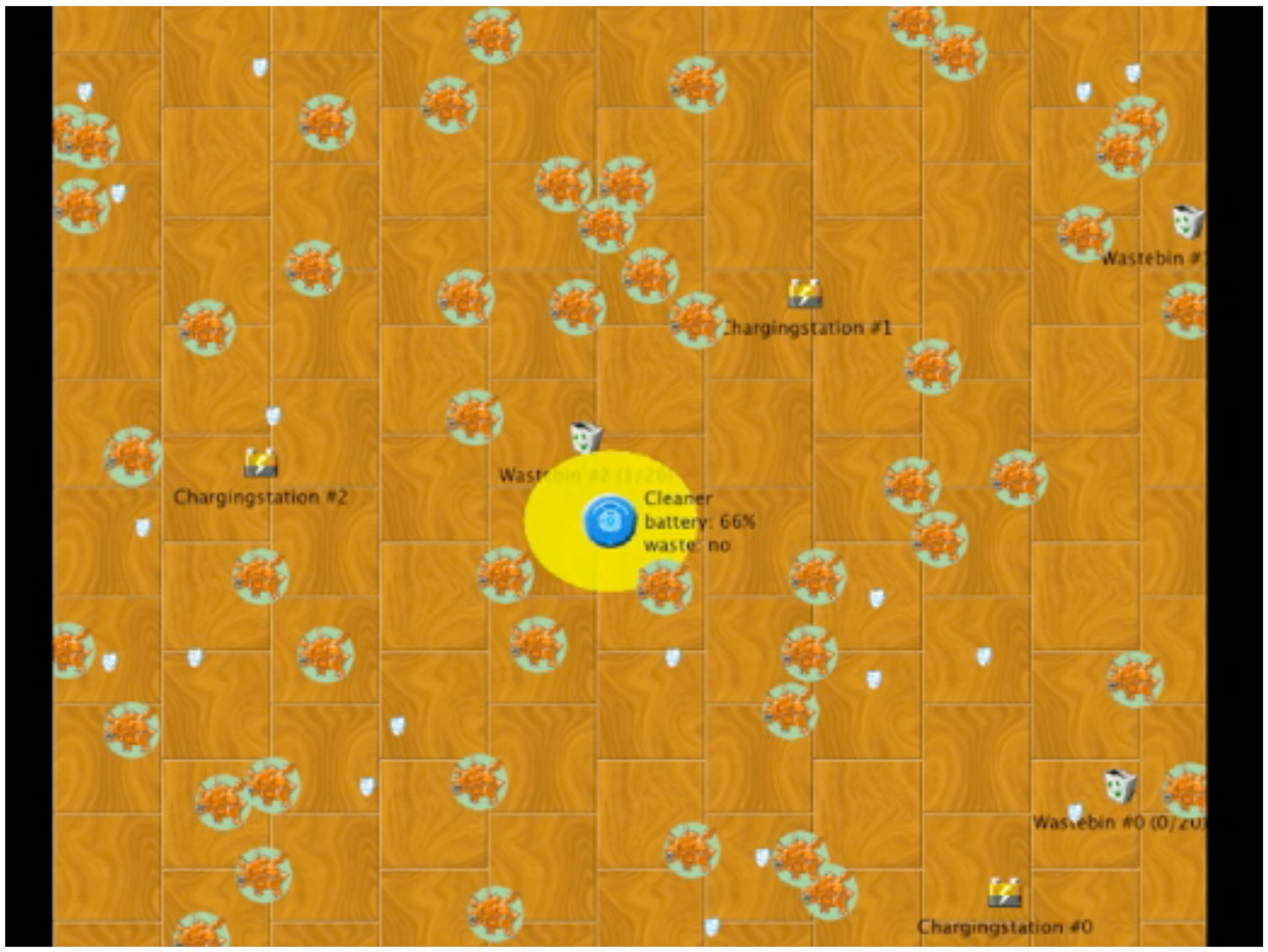
Autonomous cleaning agent

- explore locations of important objects
- look for waste and bring them to the closest bin
- maintain battery charge
- avoid obstacles by changing course when necessary
- find the shortest path to reach a specific location
- stop when no movement is possible or running out of battery

Experiments

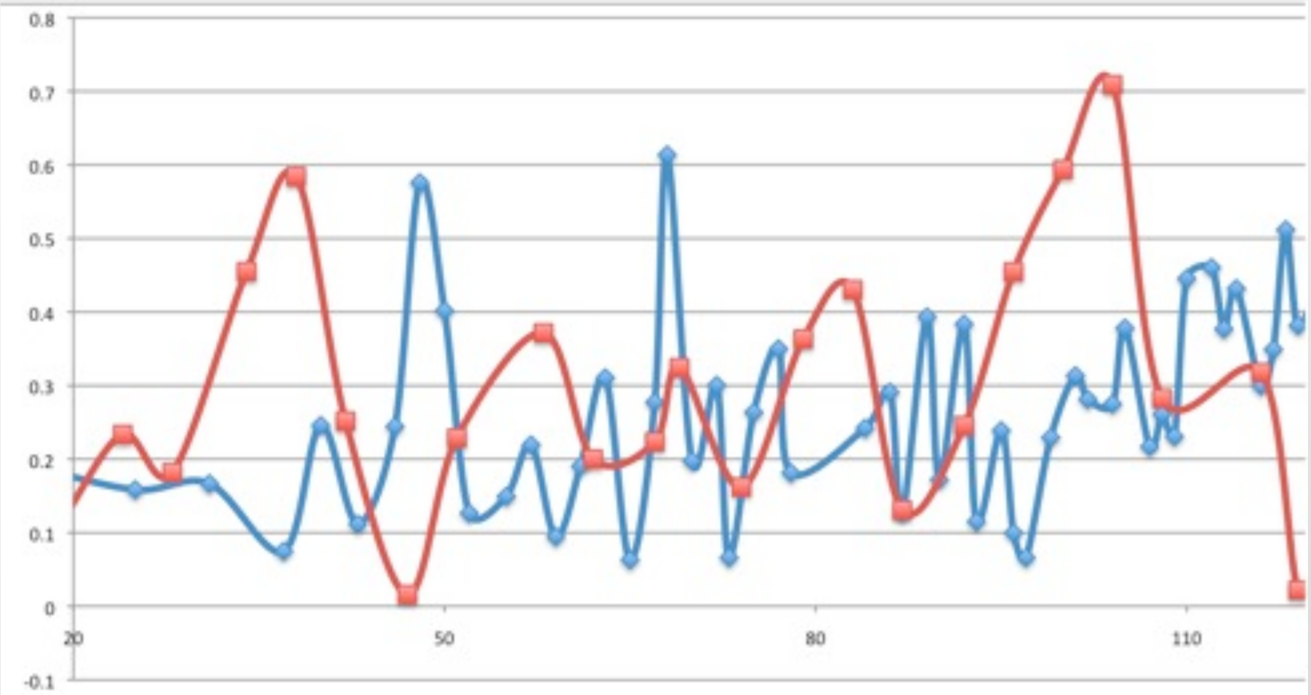
Autonomous cleaning agent

- explore locations of important objects
- look for waste and bring them to the closest bin
- maintain battery charge
- avoid obstacles by changing course when necessary
- find the shortest path to reach a specific location
- stop when no movement is possible or running out of battery



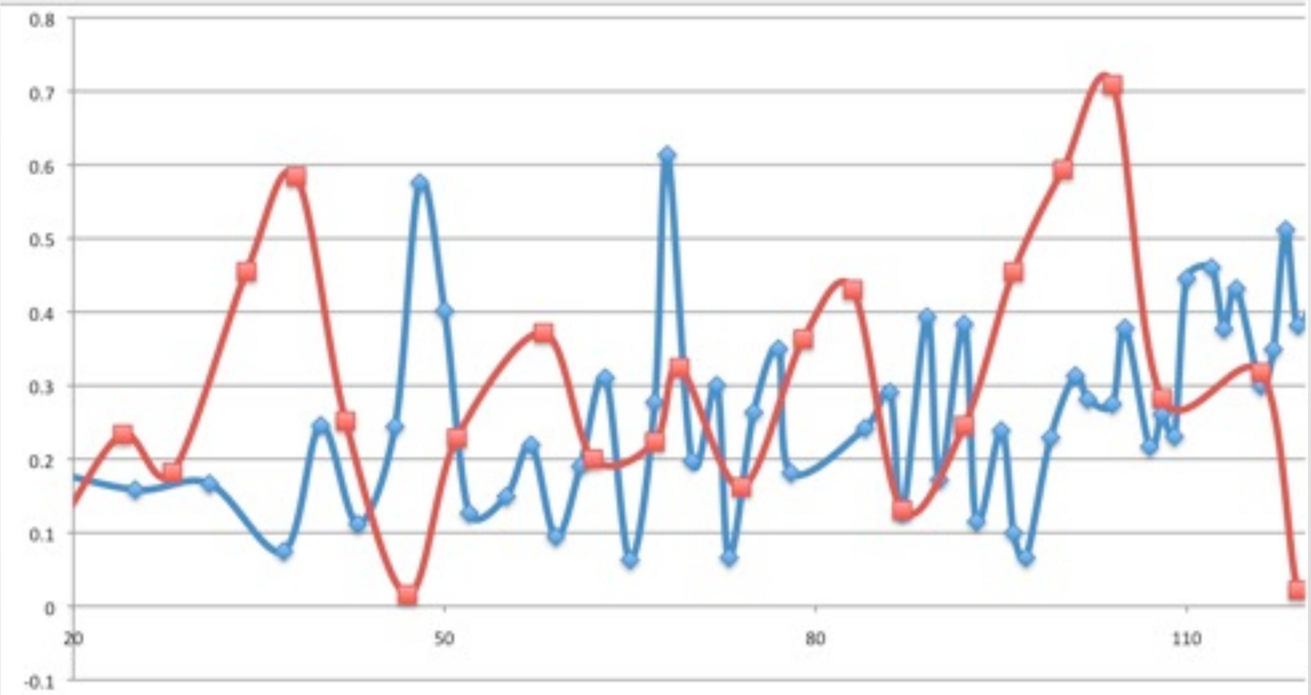
Fitness measurement

Same input environment, different outputs

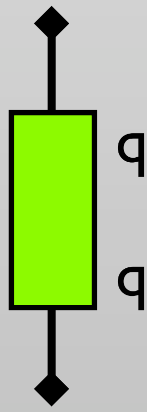


Fitness measurement

Same input environment, different outputs



Cumulative box-plots of the distance of executions converge

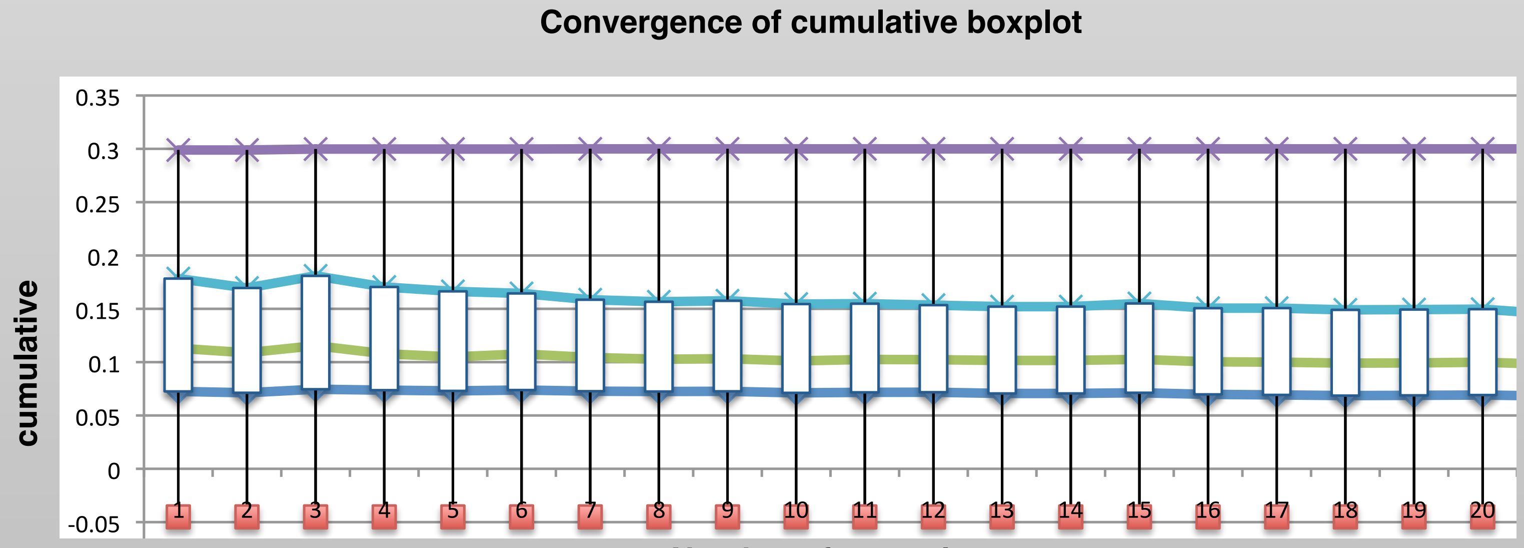


max

quartile 3 (75%)

quartile 1 (25%)

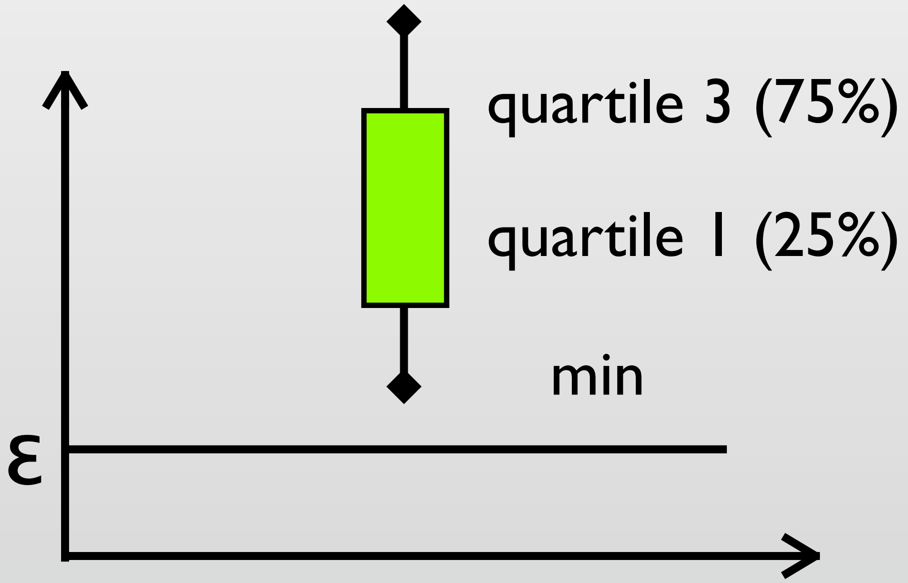
min



Fitness measurement

Fitness

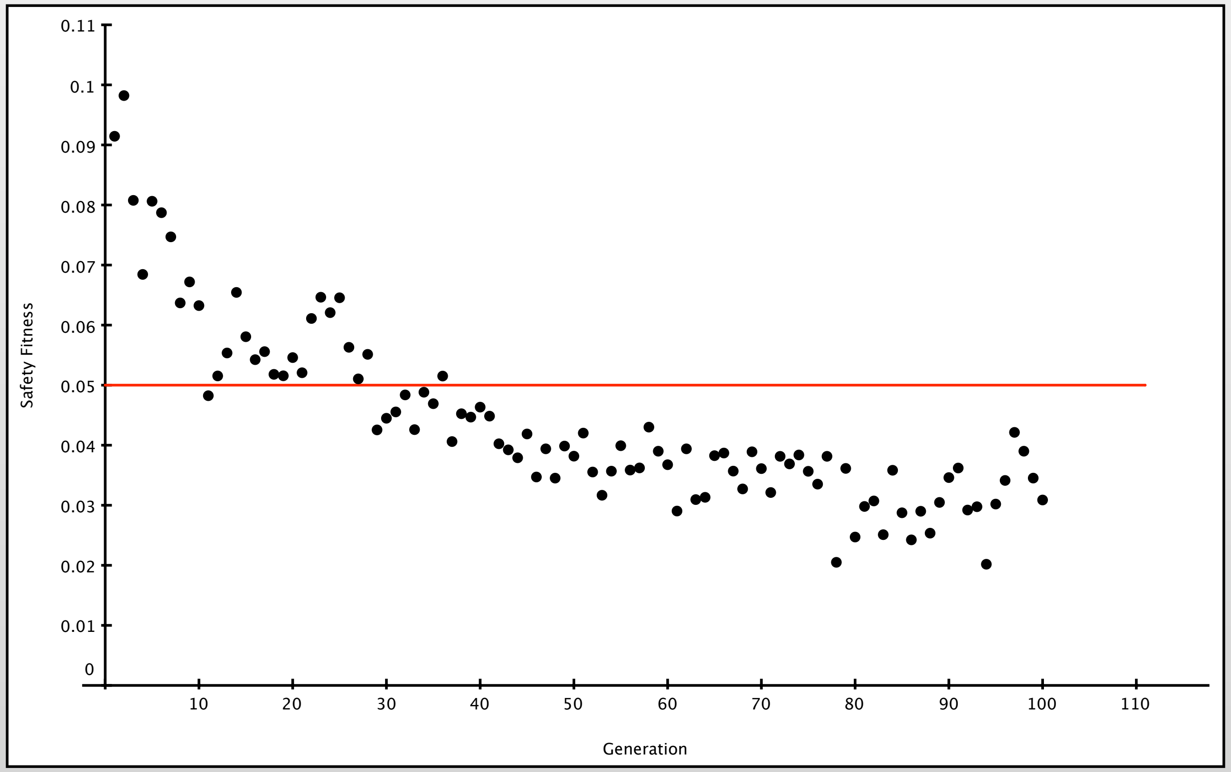
$$f = \begin{cases} min(D) + w_1 * quartile1(D) + w_3 * quartile3(D) \\ \qquad\qquad \text{if } min(D) > \varepsilon, \\ min(D) - \varepsilon \qquad \text{if } min(D) \leq \varepsilon, \\ +\infty \text{ if the agent cannot move and suspend safely.} \end{cases}$$



Search objective: *bringing the box down to the threshold* ε *,i.e. leading the agent to hit obstacles*

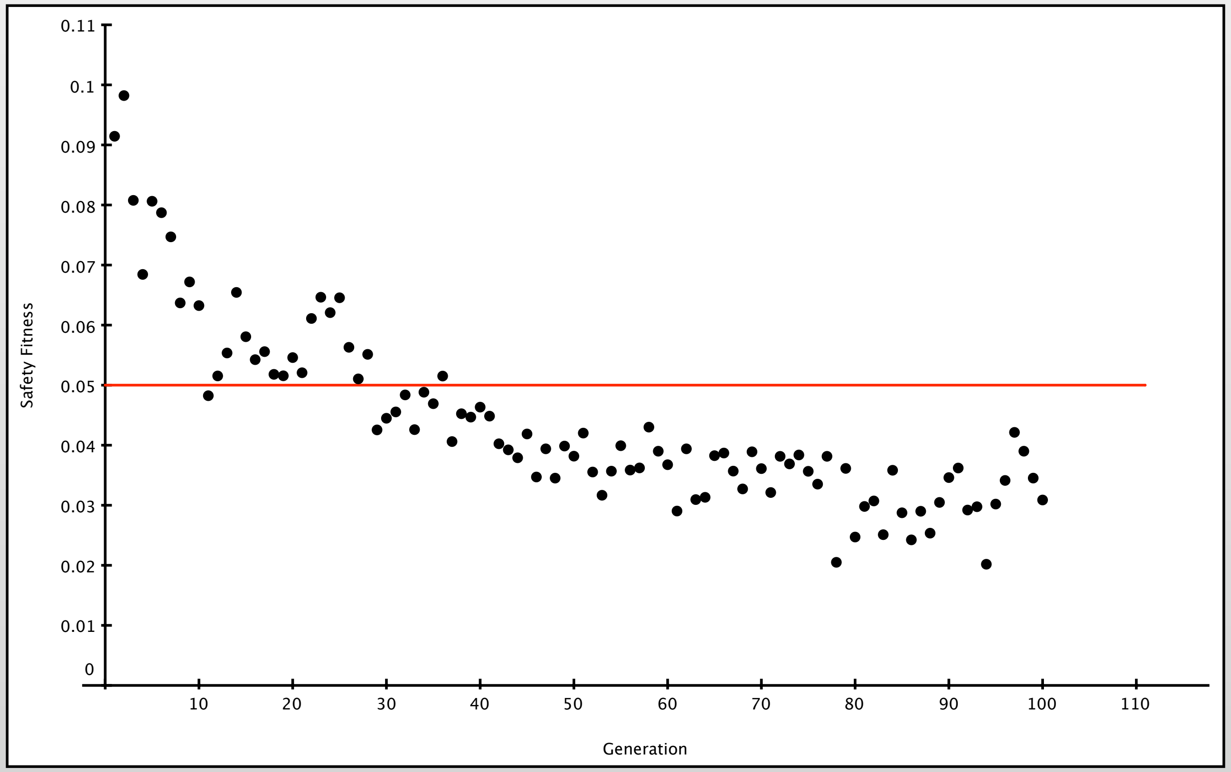
Result & discussion

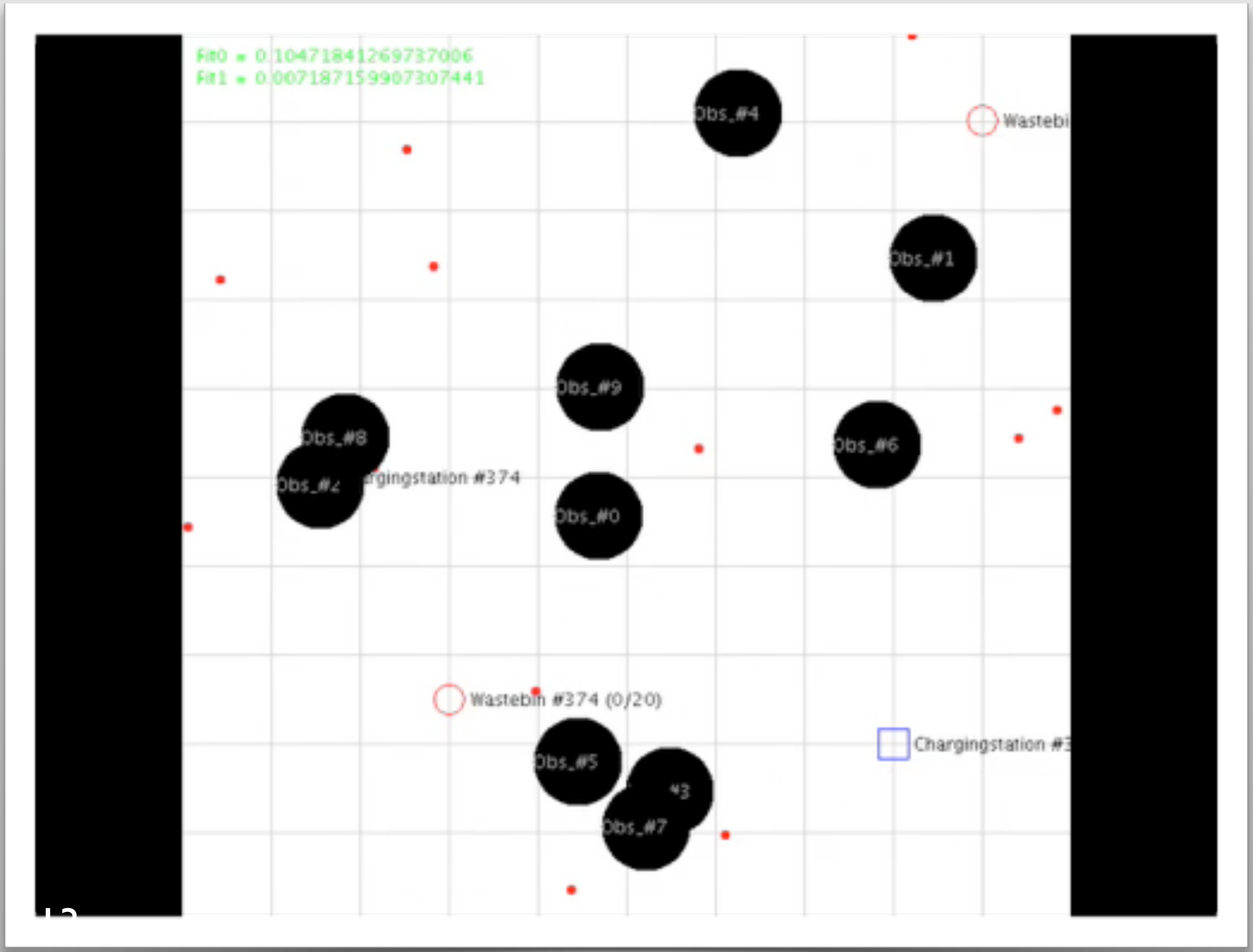
evolutionary testing



Result & discussion

evolutionary testing

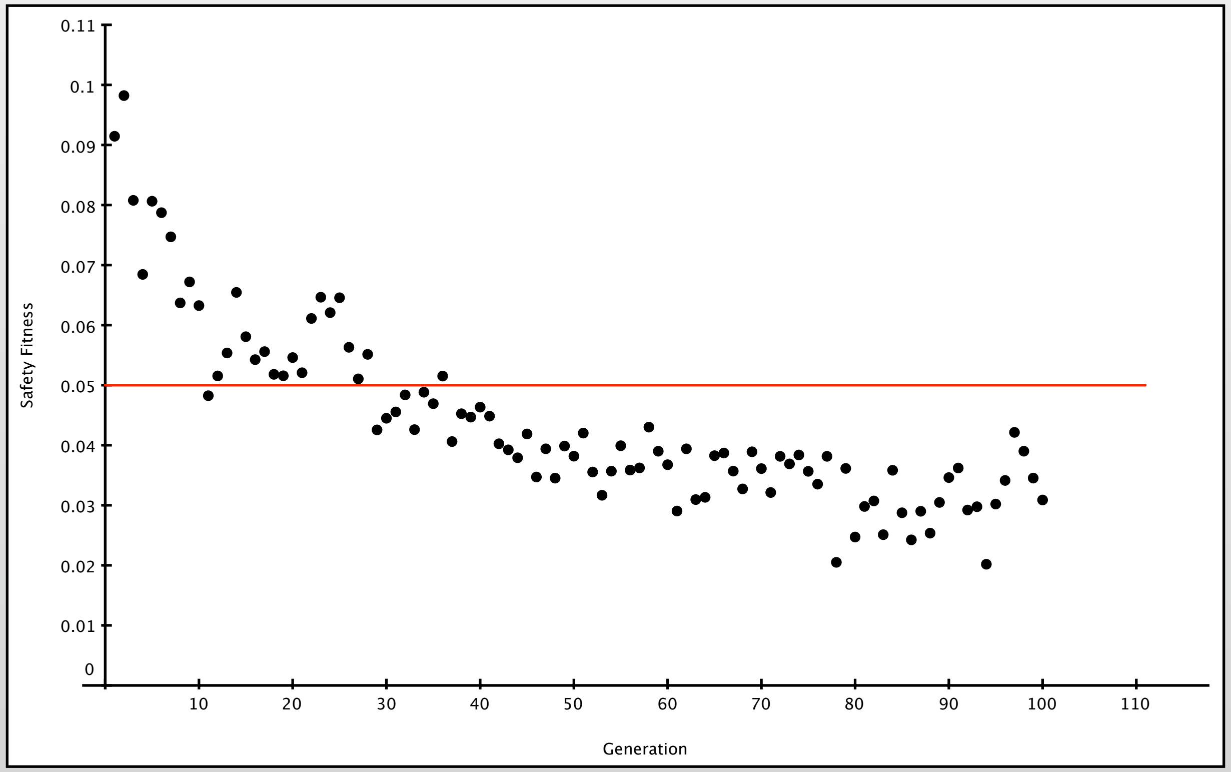




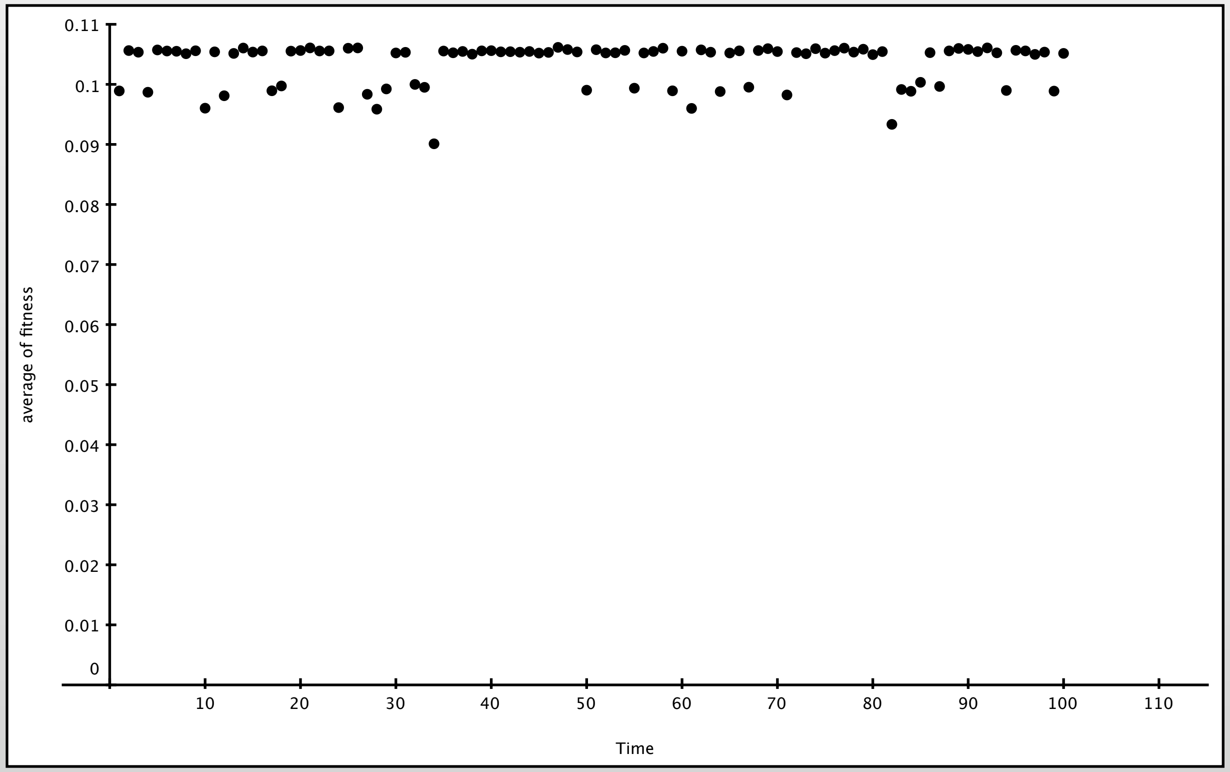
```
Fit0 = 0.10471841269737006
Fit1 = 0.0071871599073O7441
```

Result & discussion

evolutionary testing



random testing



- evolutionary testing found better test cases than random testing
- and is more effective in detecting faults

Conclusion

- Autonomous agent testing is hard
  - Non-deterministic outputs
  - Variability of the world setting
- Evolutionary testing
  - Use quality requirements as evaluation criteria
  - Use them to guide the evolutionary generation of test input
  - Is more effective compared to random testing
  - Is cost-effective, requires almost no additional cost