



Software Testing with Evolutionary Strategies

Introduction

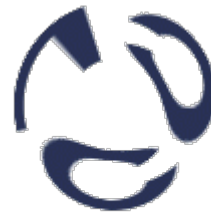
Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work



Lenguajes y Ciencias
de la Computación

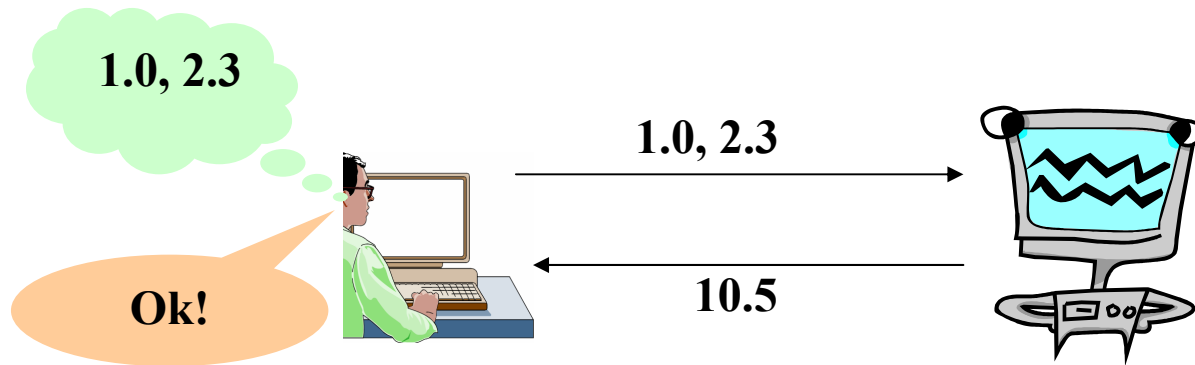


UNIVERSIDAD
DE MÁLAGA

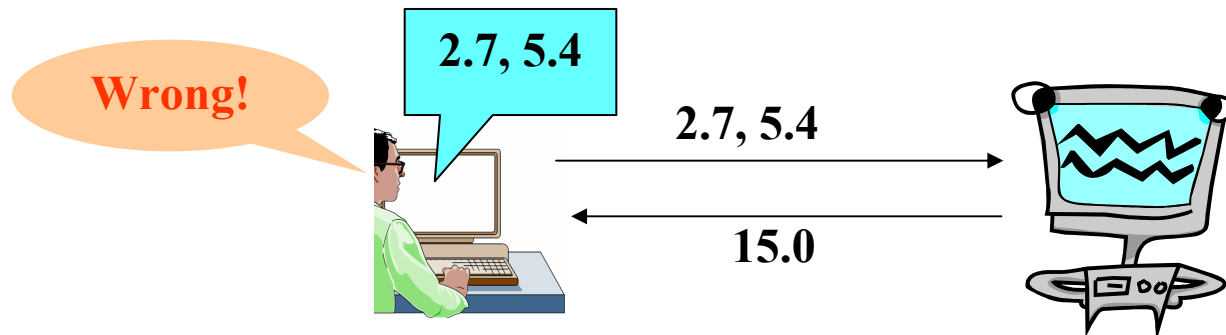
Enrique Alba and J. Francisco Chicano

Introduction

- After codification, software products require a **test phase**
- The objective is to **find errors** and to ensure **software correctness**
- Software companies dedicate **50%** of resources to this task



- We propose an automatic tool based on **Metaheuristics** to **generate the input data for the test**





Test Adequacy Criteria

- Test data generator objective: to **propose** input data finding a **maximum** number of errors

Introduction

Previous Work

**Test Data
Generator**

**Evolutionary
Strategy**

Experiments

**Conclusions &
Future Work**



Test Adequacy Criteria

- Test data generator objectives (test adequacy criteria) **DIFFICULT TO CHECK** but data finding a maximum number of errors



- Test data generator objectives (test adequacy criteria)

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

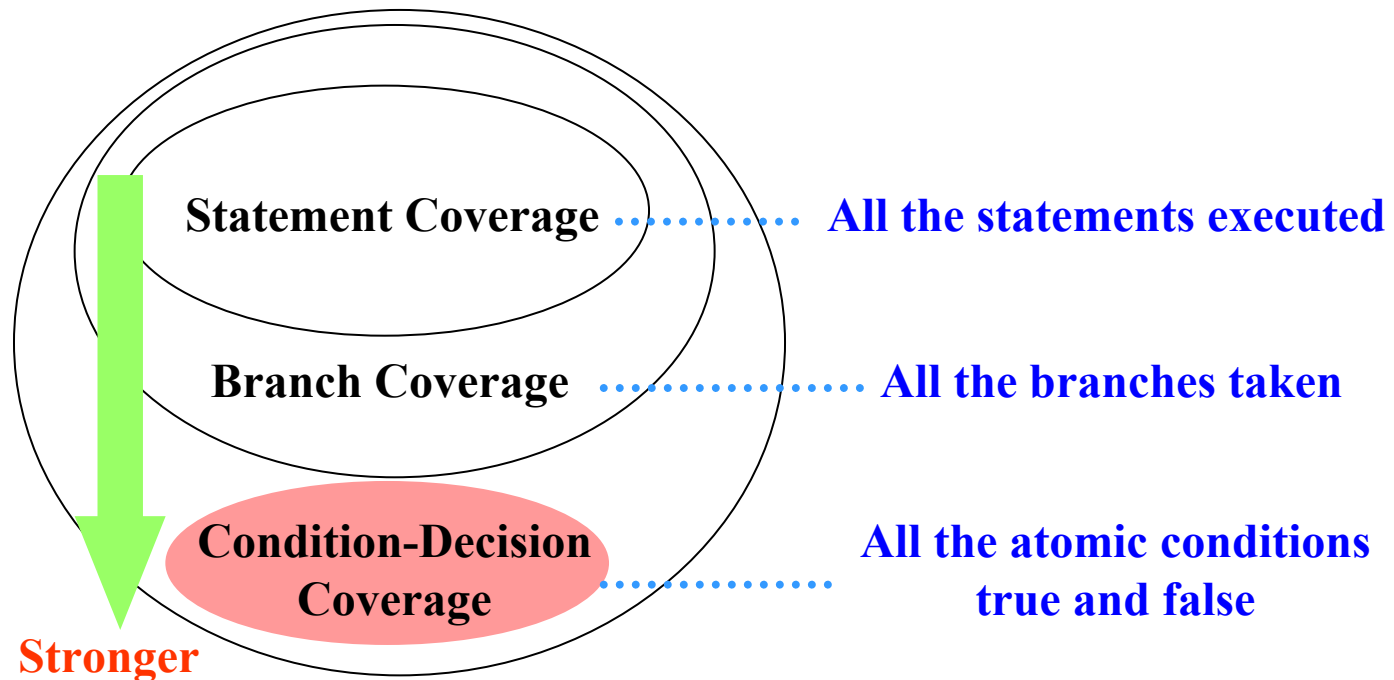
Experiments

Conclusions &
Future Work

Test Adequacy Criteria

- Test data generator objectives (test adequacy criteria)
 - Test data generator objectives (test adequacy criteria)
- DIFFICULT TO CHECK**
- maximum number of errors

- Test data generator objectives (test adequacy criteria)



Previous Work

- Three main paradigms:

- **Random** test data generation



1.2, 0.7

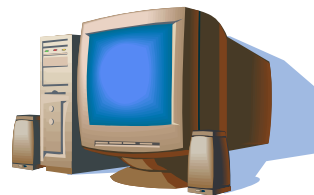
- **Symbolic** test data generation

α, β

1.0, -2.0 ← $\alpha + \beta < 0$

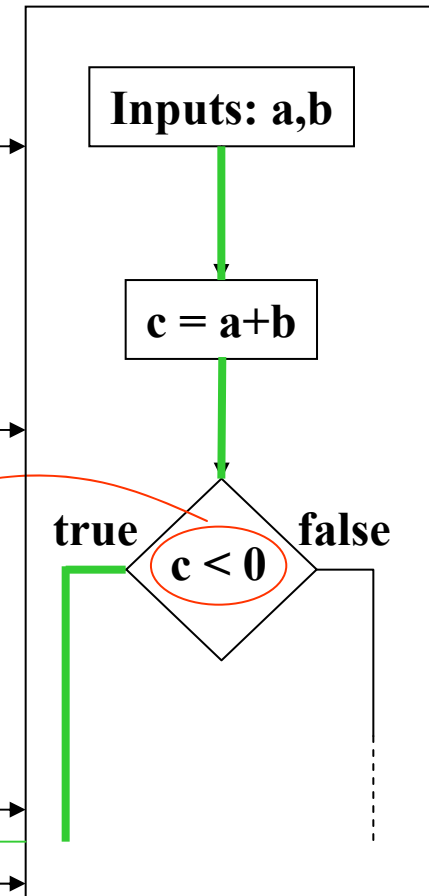
3.5, 1.2 ← $\alpha + \beta \geq 0$

- **Dynamic** test data generation



-1.0, -0.5

1.0, -0.5



Previous Work

- Three main paradigms:

- **Random** test data generation



1.2, 0.7

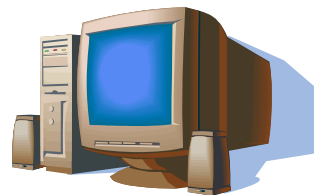
- **Symbolic** test data generation

α, β

1.0, -2.0 ← $\alpha + \beta < 0$

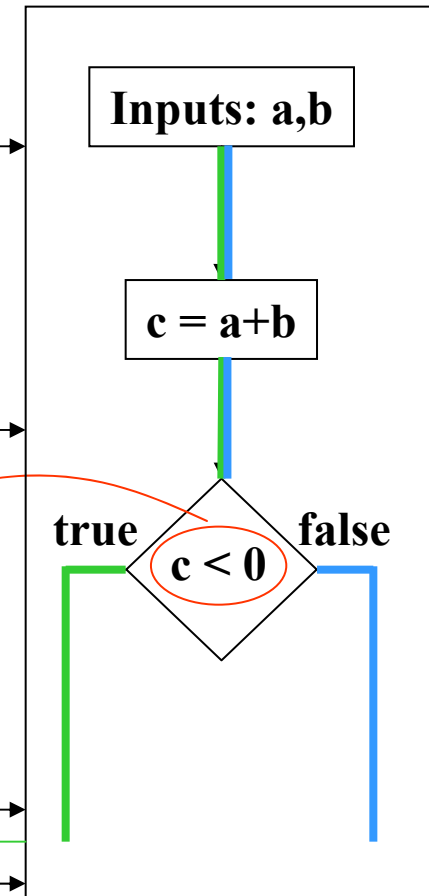
3.5, 1.2 ← $\alpha + \beta \geq 0$

- **Dynamic** test data generation



-1.0, -0.5

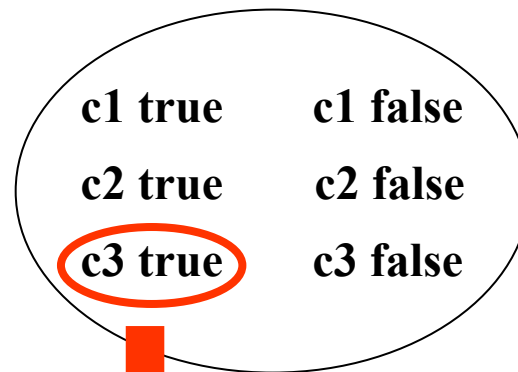
1.0, -0.5



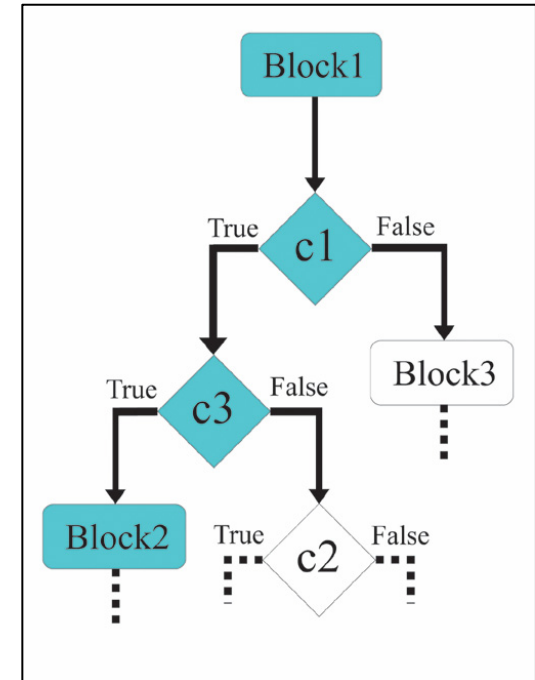
Previous Work

- The global objective is broken down in **small sub-objectives**

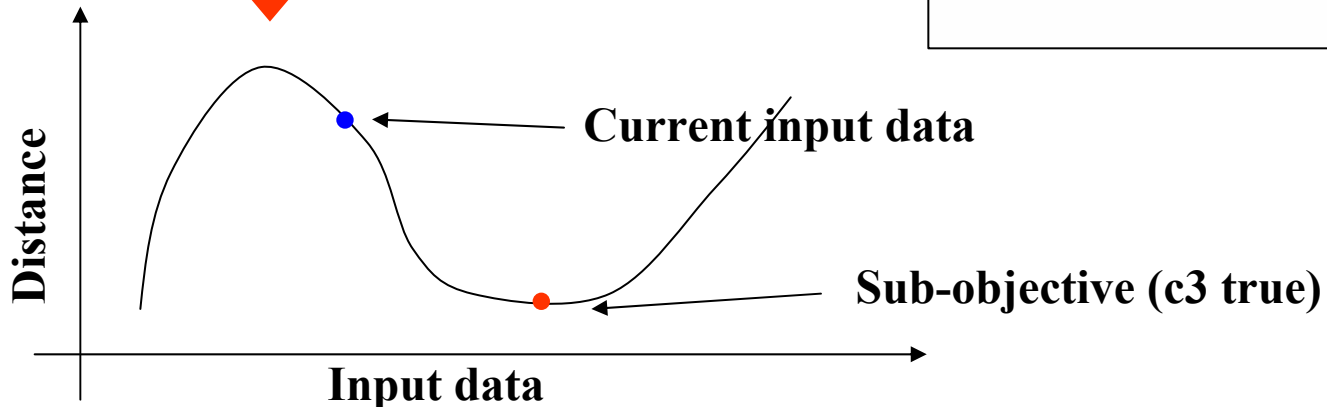
Six sub-objectives



condition-decision
coverage



Function minimization problem



Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

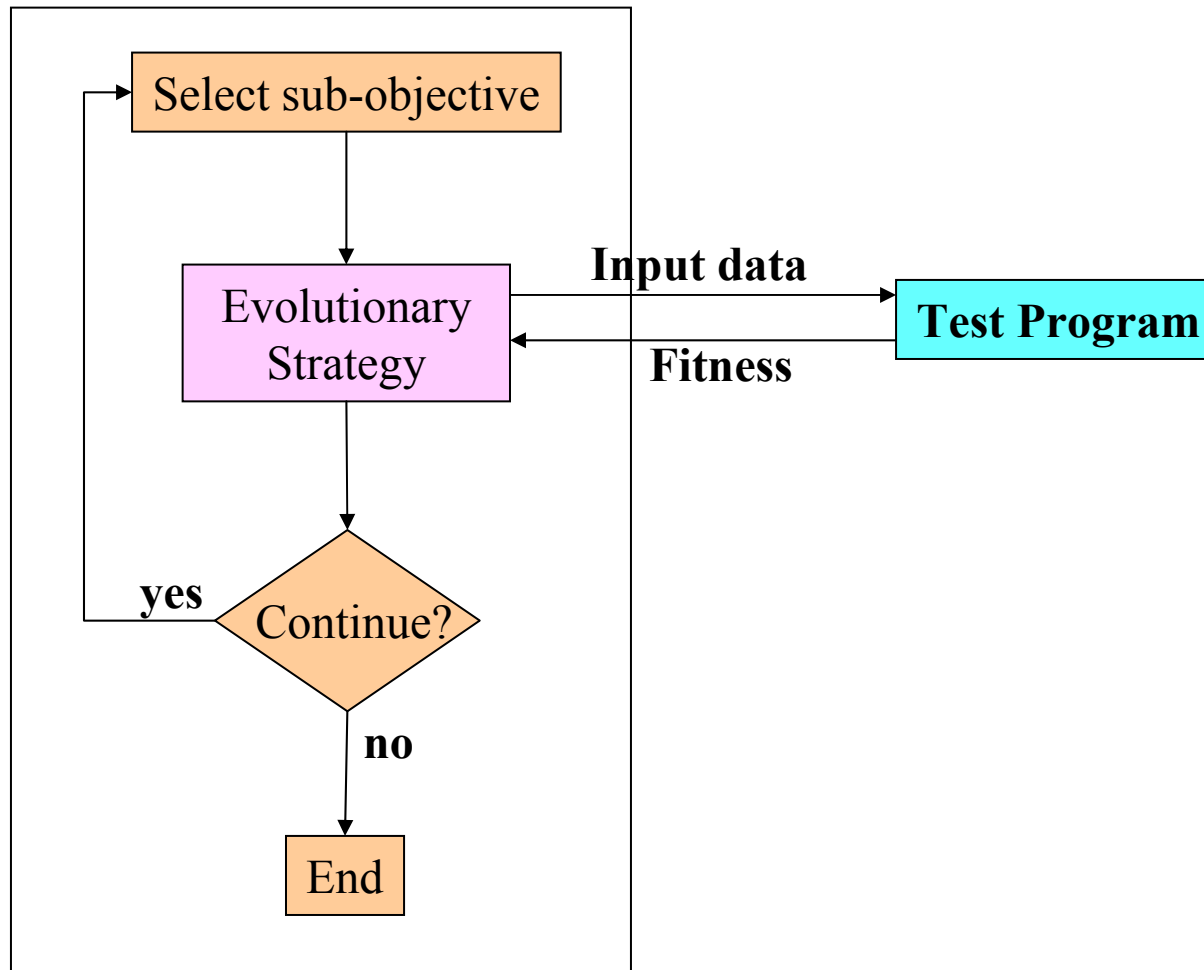
Conclusions &
Future Work



Test Data Generator



Test Data Generator



Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work

Evolutionary Strategy



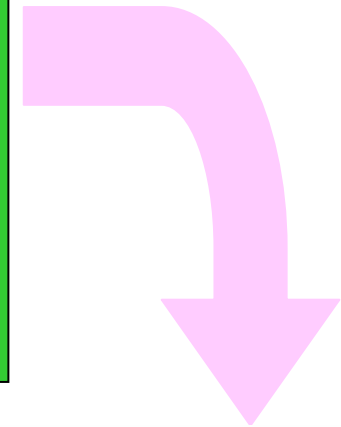
Evolutionary Algorithms

Genetic
Algorithm

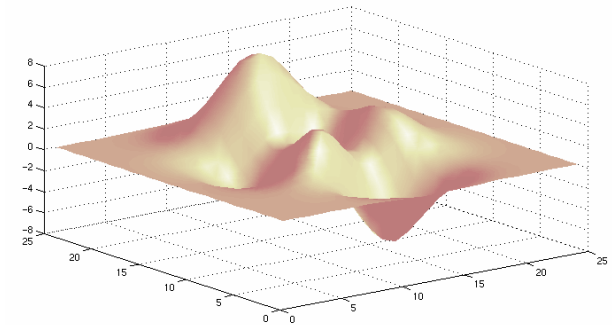
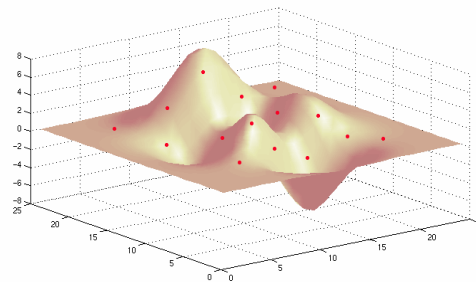
Evolutionary
Strategy

Genetic
Programming

Evolutionary
Programming



Population-based



Optimization problem

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work



Evolutionary Strategy

- Pseudo-code of an Evolutionary Strategy

```
t := 0;
P(0) := Generate();
Evaluate (P(0));
while stop criterion not met do
    P'(t) := Select (P(t));
    P''(t) := Gaussian_Mutation (P'(t));
    Evaluate(P''(t));
    P(t+1) := Replacement (P(t), P''(t));
    t := t+1;
end while;
return the best solution;
```

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work

Evolutionary Strategy

• Evolutionary Strategy

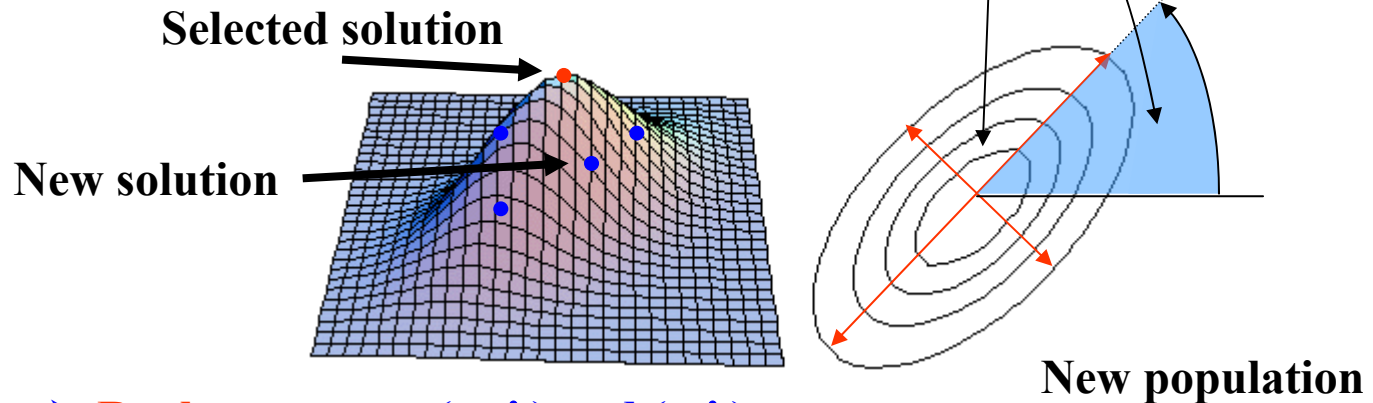
➤ Individual

(0.2, -1.4, 3.5) → Solution vector

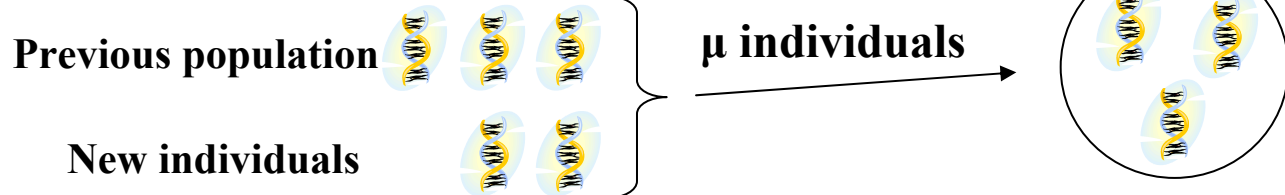
(1.0, 10.3, 7.2) → Standard deviations

(1.7, 0.3, 2.1) → Angles

➤ Gaussian Mutation



➤ Replacement → $(\mu + \lambda)$ and (μ, λ)



Evolutionary Strategy

• Evolutionary Strategy

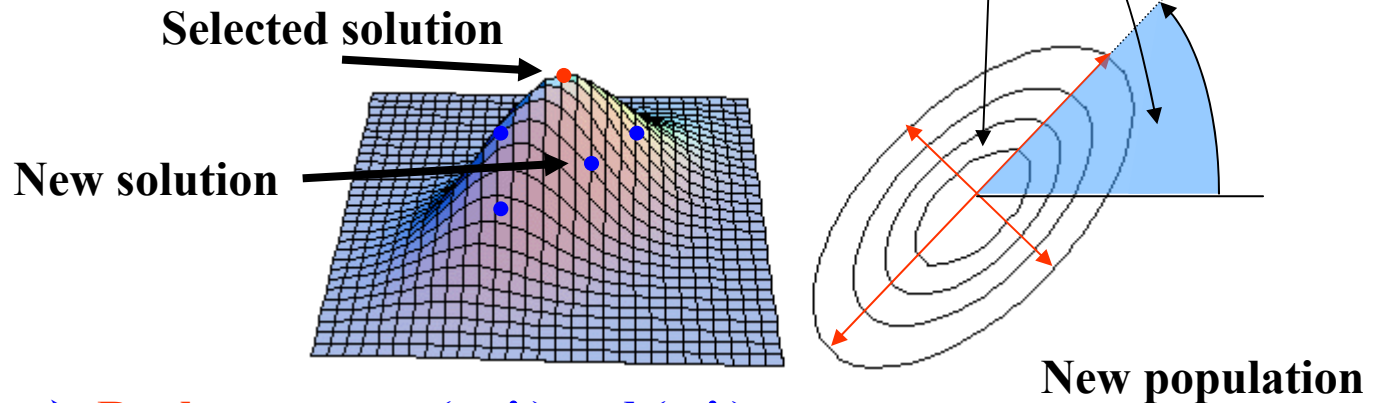
➤ Individual

(0.2, -1.4, 3.5) → Solution vector

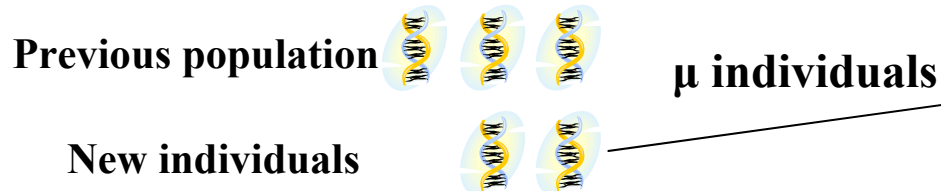
(1.0, 10.3, 7.2) → Standard deviations

(1.7, 0.3, 2.1) → Angles

➤ Gaussian Mutation



➤ Replacement → $(\mu + \lambda)$ and (μ, λ)



Introduction

Previous Work

Test Data Generator

Evolutionary Strategy

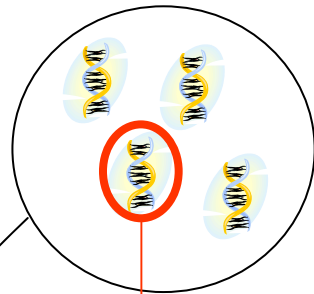
Experiments

Conclusions & Future Work

Evolutionary Strategy

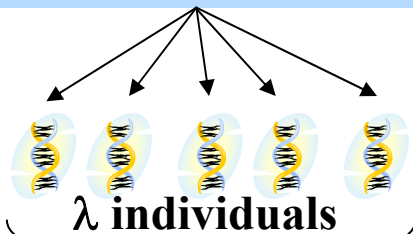


Population



μ individuals

Gaussian Mutation

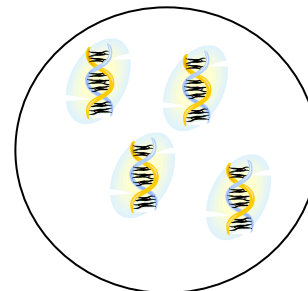


λ individuals

(0.1, -5.2, 3.0, ...)

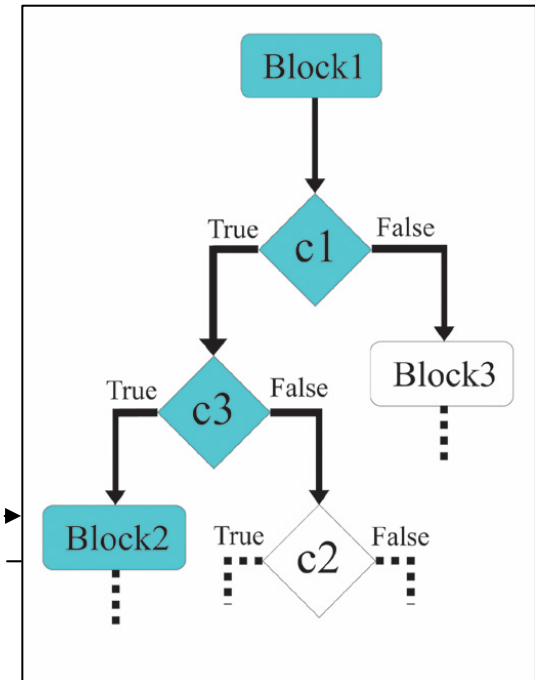
7.3

$(\mu+\lambda)$ Replacement



New population

Test Program



Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work

Experiments: Benchmark

- **Benchmark: 11 programs in C language**

Program	Conds.	LOC	Args.	Source
triangle	21	53	3	Michael et al., 2001
gcd	5	38	2	Authors
calday	11	72	3	C-Recipes
crc	9	82	13	C-Recipes
insertion	5	47	20	C-Recipes
shell	7	58	20	C-Recipes
quicksort	18	143	20	C-Recipes
heapsort	10	72	20	C-Recipes
select	28	200	21	C-Recipes
bessel	21	245	2	C-Recipes
sa	30	332	23	C-Recipes

- **We perform 30 independent runs to get statistical confidence**



- Introduction
- Previous Work
- Test Data Generator
- Evolutionary Strategy
- Experiments**
- Conclusions & Future Work

Experiments: First Results

- Algorithm: (10+1)-ES
- Maximum iterations: 100

```
while(1<2)
{
    /* something */
}
```

Code-dependent coverage loss

```
p = malloc(4);
if (!p)
{
    error();
}
```

Environment-dependent coverage loss

Program	Avg. Cov.(%)	Avg. Evals.	Avg. Time(s)
triangle	97.54	1975	10.6
gcd	100.00	21	0.0
calday	91.82	1182	6.8
crc	94.44	1114	28.7
insertion	100.00	10	0.0
shell	100.00	10	0.0
quicksort	88.89	1110	8.4
heapsort	90.00	1110	9.4
select	53.57	1120	8.9
bessel	95.08	1306	7.6
sa	97.06	1329	5082.1

- Condition-decision is not a good metric for measuring accuracy

Introduction

Previous Work

Test Data Generator

Evolutionary Strategy

Experiments

Conclusions & Future Work

Experiments: First Results

$$\text{Corrected coverage} = \frac{\text{reached sub-objectives}}{\text{reachable sub-objectives}}$$

(10+1)-ES		Corrected Coverage (%)		
Program	Cov.(%)	Avg.(%)	Max(%)	Std. Dev.
triangle	97.54	99.92	100.00	0.0044
gcd	100.00	100.00	100.00	0.0000
calday	91.82	91.82	100.00	0.0246
crc	94.44	100.00	100.00	0.0000
insertion	100.00	100.00	100.00	0.0000
shell	100.00	100.00	100.00	0.0000
quicksort	88.89	94.12	94.12	0.0000
heapsort	90.00	100.00	100.00	0.0000
select	53.57	83.33	83.33	0.0000
bessel	95.08	97.40	97.56	0.0061
sa	97.06	98.70	100.00	0.0072

Introduction

Previous Work

Test Data
GeneratorEvolutionary
Strategy

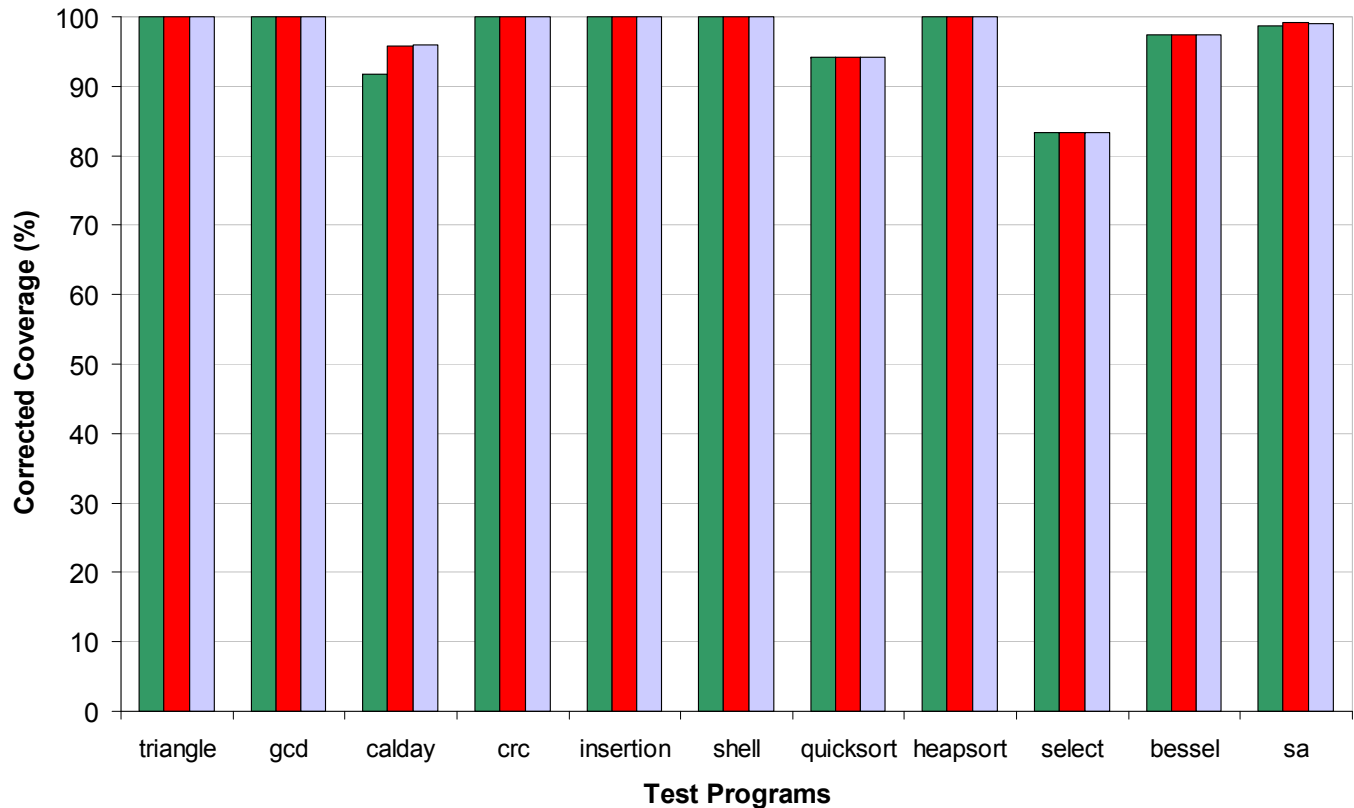
Experiments

Conclusions &
Future Work

Experiments: Influence of λ



Number of offspring: ■ 1 ■ 2 ■ 3



Influence of λ

Coverage: negligible

Number of evaluations: important

Introduction

Previous Work

Test Data
Generator

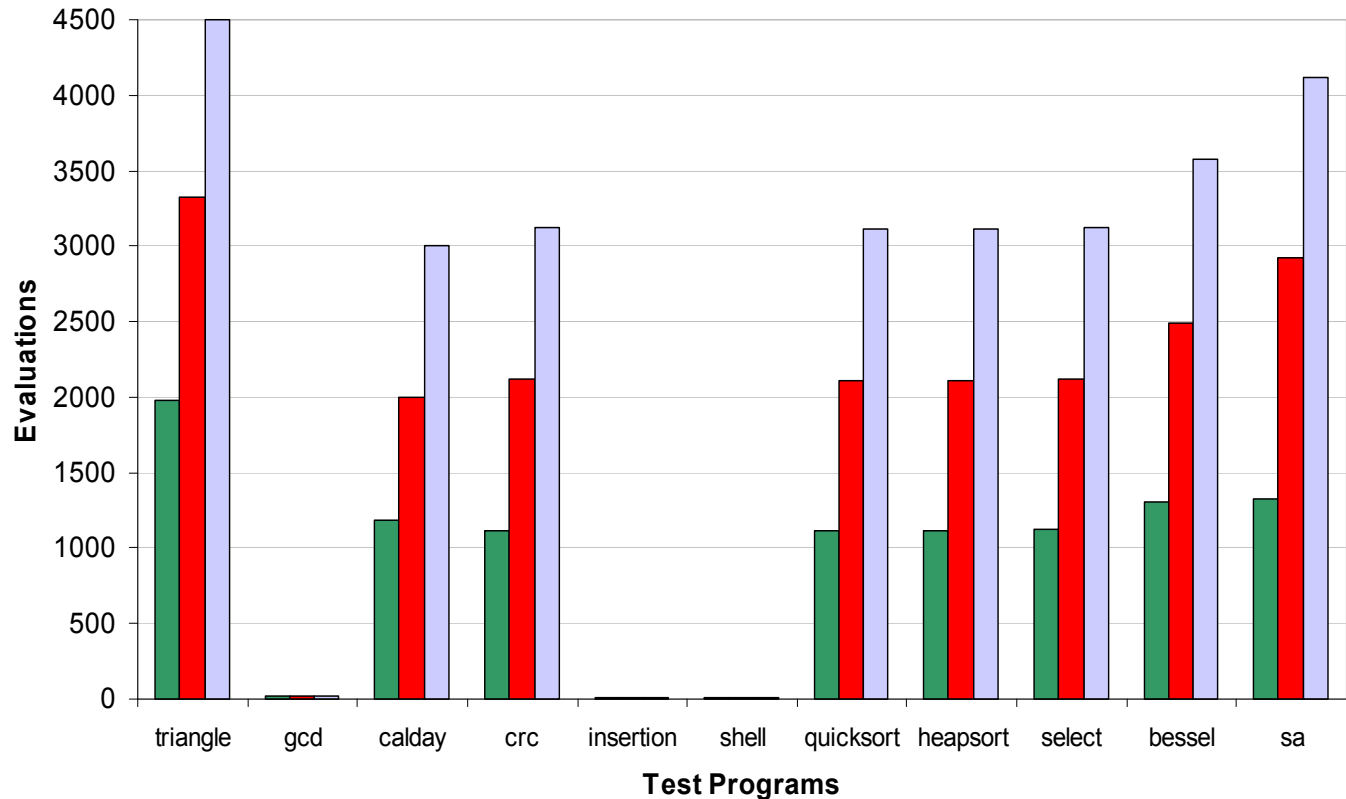
Evolutionary
Strategy

Experiments

Conclusions &
Future Work

Experiments: Influence of λ

Number of offspring: ■ 1 ■ 2 ■ 3



Influence of λ

Coverage: **negligible**

Number of evaluations: **important**

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

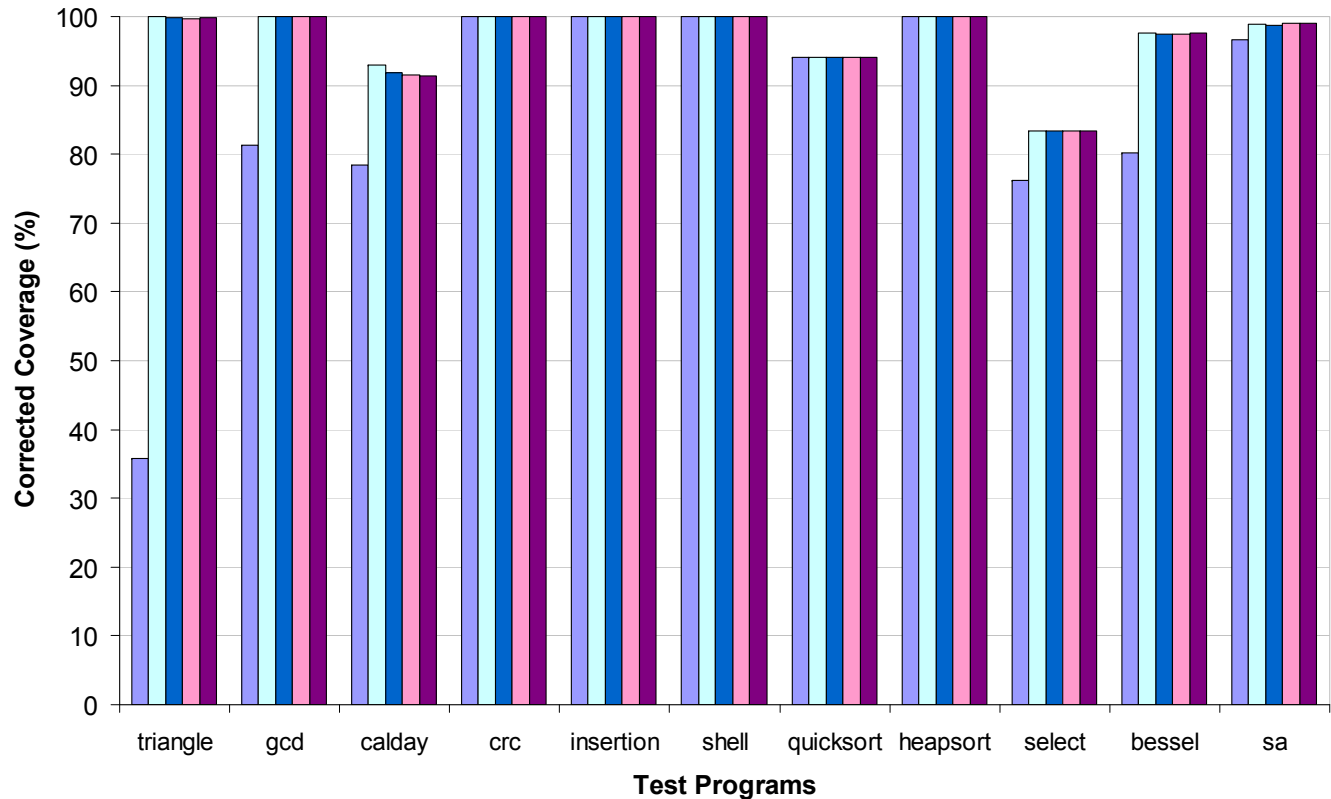
Conclusions &
Future Work



Experiments: Influence of μ



Population size: ■ 1 ■ 5 ■ 10 ■ 20 ■ 30



Influence of μ

Coverage: **negligible** for $\mu \geq 5$

Number of evaluations: **slight**

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

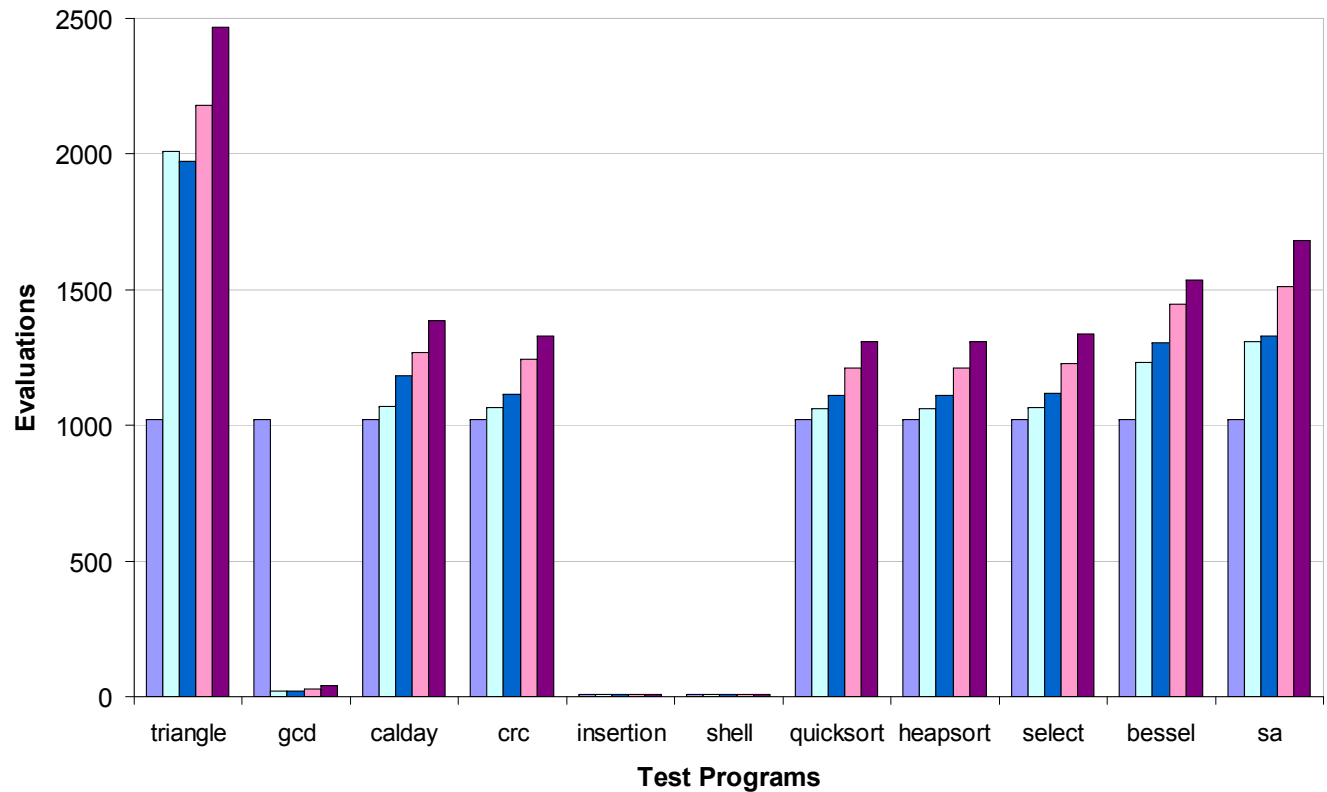
Experiments

Conclusions &
Future Work

Experiments: Influence of μ



Population size: ■ 1 ■ 5 ■ 10 ■ 20 ■ 30



Influence of μ

Coverage: negligible for $\mu \geq 5$

Number of evaluations: slight

Introduction

Previous Work

Test Data
Generator

Evolutionary
Strategy

Experiments

Conclusions &
Future Work

Experiments: ES vs GA

Program	(10+1)-ES		(10+1)-GA	
	Avg. Cov.(%)	Avg. Evals.	Avg. Cov.(%)	Avg. Evals.
triangle	99.92	1975	95.20	2009
gcd	100.00	21	100.00	252
calday	91.82	1182	90.91	1217
crc	100.00	1114	100.00	1121
insertion	100.00	10	100.00	10
shell	100.00	10	100.00	10
quicksort	94.12	1110	94.12	1110
heapsort	100.00	1110	100.00	1110
select	83.33	1120	83.33	1339
bessel	97.40	1306	96.91	1557
sa	98.70	1329	96.61	1110

- **ES outperforms the results of the Genetic Algorithm**



Introduction

Previous Work

Test Data
GeneratorEvolutionary
Strategy

Experiments

Conclusions &
Future Work

Experiments: Previous Results

- Two main issues to compare against different works:
 - The **implementation** → <http://tracer.lcc.uma.es/problems/testing>
 - The coverage **measures**

Corrected condition-decision coverage
 ↓
Branch coverage
 ↙ ↘
DeepCover coverage ↘

triangle	Jones et al., 1996	Michael et al., 2001	Sagarna et al., 2003	Sagarna et al., 2005	(5+1)-ES (this work)
Coverage(%)	100.00	94.29	100.00	100.00	100.00
Evaluations	18000	≈8000	608	3439	2010

- The (5+1)-ES gets **total coverage with less number of evaluations** (exception: Sagarna et al., 2003)



- Introduction
- Previous Work
- Test Data Generator
- Evolutionary Strategy
- Experiments**
- Conclusions & Future Work



Conclusions & Future Work

Conclusions

- We propose a **test data generator** using an **ES** for software testing
- Condition-decision coverage **is not a good measure** of the generator accuracy → **Corrected condition-decision coverage**
- Increasing the number of offspring has a **negligible** impact in the **coverage** but it **increases the number of evaluations** needed
- The population size must be **larger than one** for a good coverage
- The **ES-based test data generator** **outperforms** the results of the **GA-based one**

Future Work

- Apply the test data generator to **telecommunication software**
- Use **parallel evolutionary strategies**
- Integrate some **static analysis techniques**

THE END



Thanks for your attention !!!



Introduction

Previous Work

**Test Data
Generator**

**Evolutionary
Strategy**

Experiments

**Conclusions &
Future Work**