UNIVERSITÄT PADERBORN

Software Engineering

# Landscape depending parameter tuning for search-based software testing

# Overview

1. Introduction
2. Fundamentals
3. Experimental
4. Adaptive parameter control
5. Evaluation
6. Conclusion

1. Introduction

2. Fundamentals

3. Experimental

4. Adaptive parameter control

5. Evaluation

6. Conclusion

# Introduction

- Unit tests
- maximize coverage (line, branch, exception)
- lack of sufficient tests
- costly and time-consuming
- => use search-based software testing

# Motivation

- Tools... => EvoSuite state-of-the-art
- may not terminate => search budget
- optimal only with optimal configuration
- No Free Lunch theorem
  - impossible to find optimal configuration for all problems
- EvoSuite's default configuration is fairly good, but not perfect

# Research goal

- wide variety of problem-cases
- concept landscape depending
- adaptive parameter control
- increase coverage of EvoSuite

# State-of-the-art

# Challenges

■

# Delimitation

- 

1. Introduction
2. Fundamentals
3. Experimental
4. Adaptive parameter control
5. Evaluation
6. Conclusion

# Search-based software testing

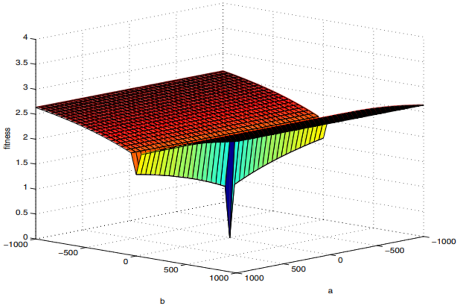
- tests for object oriented languages
- sequence of method calls
-
-
-

# Fitness function

- 
- function for e.g. coverage
- guidance for search algorithms
- 
- 

```
void bar(int x) {
    if (x == 1) {
        // uncovered code
    }
}
```

# Fitness landscape



# Genetic algorithm

### Heading

- start with random population
- iterate till termination condition
    - mutate and crossover
- return last generation

### Heading

- 
- 
- 

# DynaMOSA

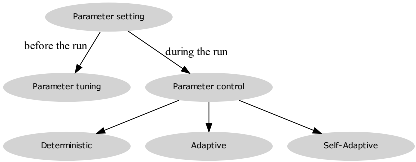
**Heading**

- start with random population
- multiple target
- keep track of target covering individuals

**Heading**

- iterate till termination condition
  - breed offspring
  - update targets
  - update archive
  - select by rank
- return archive as last generation

# Parameter tuning and control



1. Introduction

2. Fundamentals

3. Experimental

4. Adaptive parameter control

5. Evaluation

6. Conclusion

# Corpus

**SF110**

- 110 open-source Java projects
- 23,894 Java Classes
- 
- 
- 

**Panichella et al.**

- 117 open-source Java projects
- 346 Java Classes
- non-trivial and complex
- often used

# Prediction sample

- $S_1$
- 709 Java Classes
- randomly selected
- SF110

- 9.8 days on three machines
- 
- 
- 

# Evaluation sample

- $S_2$
- 346 Java Classes
- the whole Panichella corpus
- 
- 4.8 days on tree machines
- 
- 
- 

# Sensitive sample

- $S_3$
- 20 Java Classes
- extracted from $S_1$
- high Standard Deviation

- 6.6 hours on tree machines
-
-
-

# Comparisons

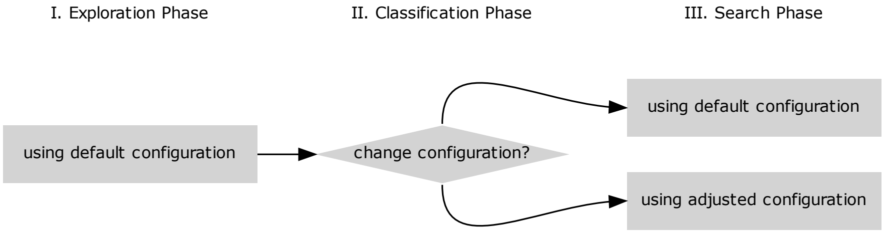
- 30 repeats for every Java Class
- Mann-Whitney U-test
- Vargha-Delaney effect size $\hat{A}_{12}$

1. Introduction
2. Fundamentals
3. Experimental
4. Adaptive parameter control
5. Evaluation
6. Conclusion

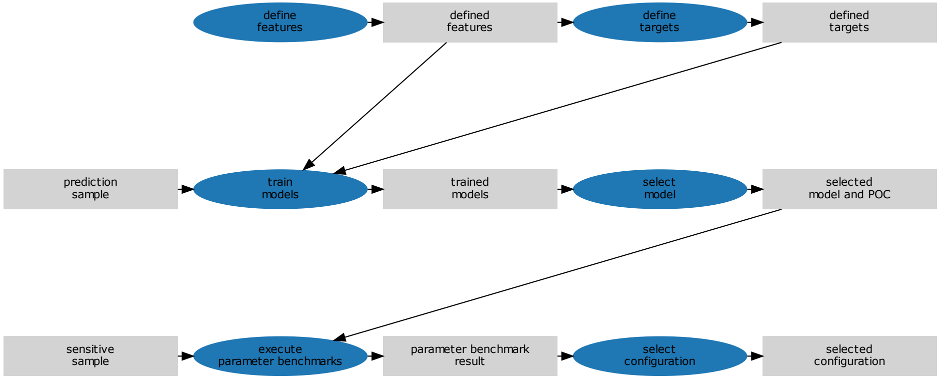
# Concept

- use the beginning of the search to explore the search space
- classify the current search (”adjust” or ”default”)
- use configuration depending on class

# Configuration selection

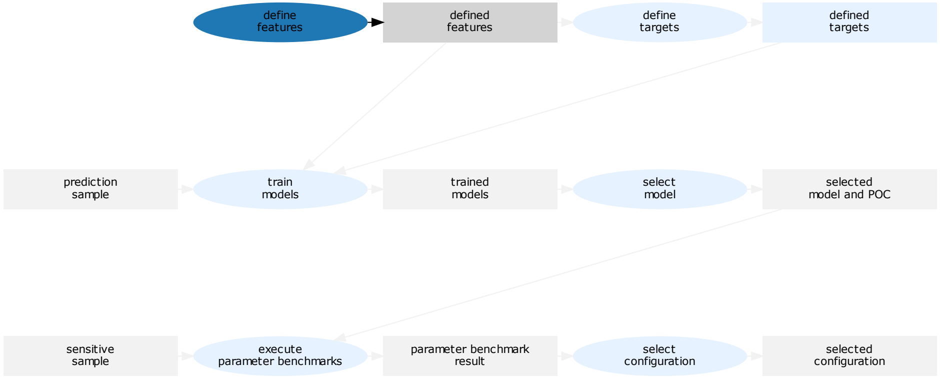


# Components



define
features
defined
features
define
targets
defined
targets
prediction
sample
train
models
trained
models
select
model
selected
model and POC
sensitive
sample
execute
parameter benchmarks
parameter benchmark
result
select
configuration
selected
configuration

# Landscape features



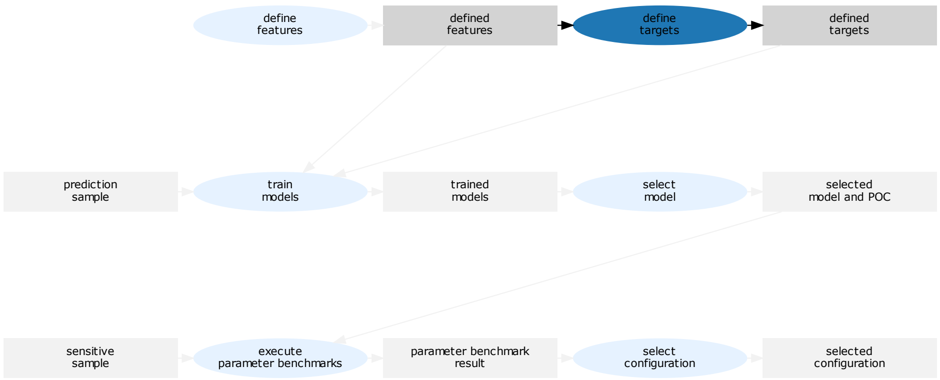
# Landscape approximation

- random walk is time-consuming
- instead: using the first generations
- ”long jump” instead a step [Kauffman and Levin, 1987]
- approximation of the landscape

# Landscape features

- Fitness and gradient
- Neutrality
- Neutrality Volume
- Information Content

# Classification target



# Target approximation

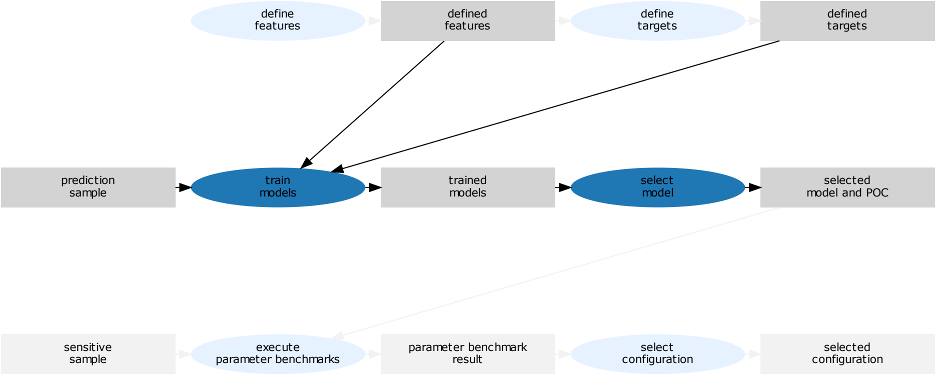
# Targets

- Low end coverage $endcov. < 0.8$
- High standard deviation $stdev > 0.1$
- Relative low coverage $cov. < max(cov.) * 0.8$

# Targets

| target | n in class | description |
|---|---|---|
| $cov. < max(cov.) * 0.8$ | 447 | End coverage less than 80% of the best execution |
| $endcov. < 0.8$ | 6687 | End coverage less than 80% |
| $stdev > 0.1$ | 541 | Standard deviation greater than 0.1 |
| $stdev > 0$ | 6046 | Standard deviation greater than 0.0 |
| $p = 125$ | 1735 | The median coverage with a population of 125 is greater than the median coverage with default settings |
| $stdev > 0.1$ & $cov. < max(cov.) * 0.8$ | 256 | The boolean ”and” of the two targets |
| $stdev > 0$ & $cov. < max(cov.) * 0.8$ | 447 | The boolean ”and” of the two targets |
| $stdev > 0.1$ & $endcov. < 0.8$ | 391 | The boolean ”and” of the two targets |
| $pop = 125$ & $cov. < max(cov.) * 0.8$ | 254 | The boolean ”and” of the two targets |

# Classification



# Classification

## Heading

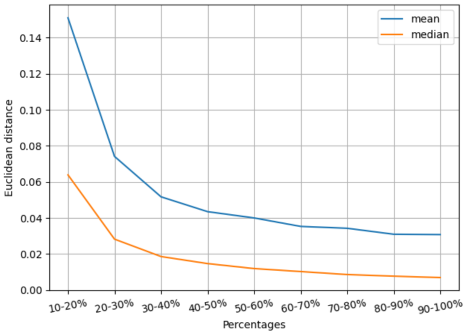
- 
- 
- 

## Heading

- 
- 
- 

# Length of Exploration Phase

- compare landscape features ever 10%
- using Euclidean distance
- trade-off between
  - time for exploration
  - time for adjusted configuration
- between 20 and 40% from search (percentage of classification)



# Machine learning algorithm

### Heading

- supervised learning
- fast classification
- few features
- decision tree

### Heading

-
-
-

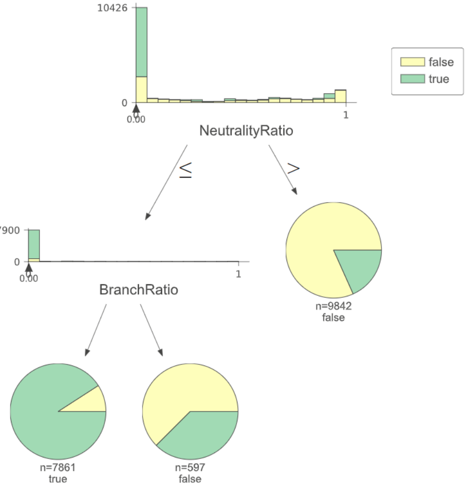
# Hyper-parameter search

- only apply on as many as possible
  - TPR > 80%
- only apply if positive effect
  - FPR < 5%
- percentage of classification (POC)
- construction criteria (Gini Impurity, Entropy or Log-Loss)
- depth of decision tree

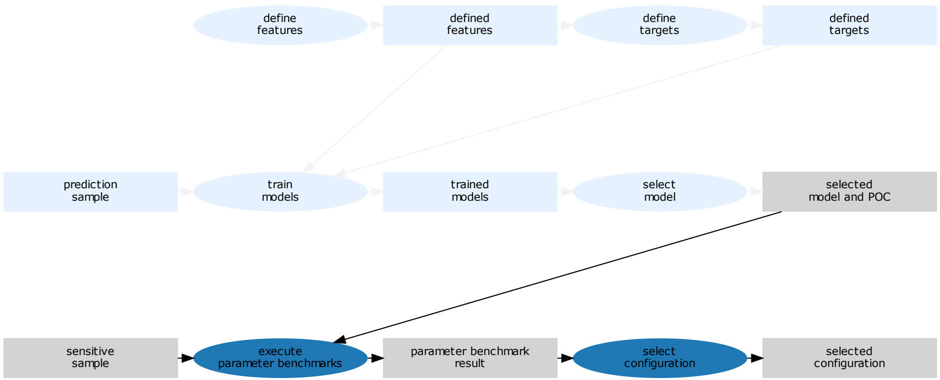
# Component selection

## Decision tree

- $stdev > 0.1 \& cov.max(cov.) * 0.8$
- depth of decision tree: 2
- Gini Impurity
- POC: 30%



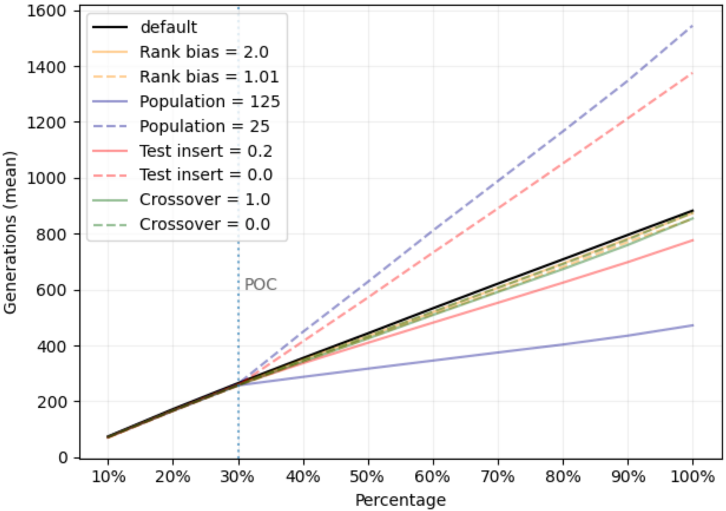
# Parameter selection



# Parameter selection

- many EvoSuite parameter
- nearly infinite possible combinations
- small-scale experiments
- using small sample $S_3$

# Influence on runtime



# Event probability

- runtime in relation to event probability
    - insert
    - mutate
    - crossover
- trade-off
    - fitness evaluation
    - event occurrence

$$P_{adj}(e) = \frac{n_1 P_1(e) + n_2 P_2(e)}{n_1 + n_2} \tag{1}$$

$$\hat{P}(e) = \sum_{i=k}^{n} \frac{n!}{i!(n-i)!} P_{adj}(e)^i (1 - P_{adj}(e))^{n-i} \tag{2}$$

# Component selection

| configuration | gen. | n | $P_{adj}$ mut. | cross. | ins. | $\hat{P}$ for $k = 5$ mut. | cross. | ins. | $n\_+$ | $n\_-$ |
|---|---|---|---|---|---|---|---|---|---|---|
| default | 853 | 9 | 0.95 | 0.68 | 0.05 | 1.0 | 0.87 | 0.0 | | |
| p = 25 | 1,263 | 13 | 0.96 | 0.69 | 0.04 | 1.0 | 1.0 $\nearrow$ | 0.0 | 0 | 0 |
| p = 125 | 572 | 6 | 0.95 | 0.68 | 0.04 | 0.77 | 0.15 | 0.0 | 3 | 0 |
| cr = 0.0 | 841 | 8 | 0.95 | 0.0 | 0.05 | 1.0 | 0.0 | 0.0 | 1 | 0 |
| cr = 1.0 | 849 | 8 | 0.95 | 0.95 | 0.05 | 1.0 | 1.0 $\nearrow$ | 0.0 | 0 | 0 |
| bias = 2.0 | 912 | 9 | 0.95 | 0.68 | 0.05 | 1.0 | 0.88 $\nearrow$ | 0.0 | 1 | 0 |
| bias = 1.01 | 824 | 8 | 0.95 | 0.68 | 0.05 | 1.0 | 0.77 | 0.0 | 0 | 0 |
| pti = 0.0 | 1,154 | 12 | 1.0 | 0.75 | 0.0 | 1.0 | 1.0 $\nearrow$ | 0.0 | 0 | 4 |
| pti = 0.2 | 770 | 8 | 0.92 | 0.63 | 0.08 | 1.0 | 0.66 | 0.0 | 0 | 0 |
| p = 10, pti = 5.0 | 912 | 9 | 0.58 | 0.13 | 0.42 | 0.69 | 0.0 | 0.31 $\nearrow$ | 3 | 0 |
| **p = 25, pti = 1.0** | **777** | **8** | **0.75** | **0.38** | **0.25** | **0.89** | **0.14** | **0.03** $\nearrow$ | **4** | **0** |

1. Introduction

2. Fundamentals

3. Experimental

4. Adaptive parameter control

5. Evaluation

6. Conclusion

# Classification

**Known sample $S_1$**

- trained on $S_1$
- test on $S_1$
- TPR of 86%
- FPR of 8%
- acceptable reliability

**Unknown sample $S_2$**

- trained on $S_1$
- test on $S_2$
- TPR of 21%
- FPR of 7%
- not reliability

# Adaptive parameter control

**Known sample $S_1$**

- 

**Unknown sample $S_2$**

- 

# Selected configuration

- 

# Discussion

- 

1. Introduction

2. Fundamentals

3. Experimental

4. Adaptive parameter control

5. Evaluation

6. Conclusion

# Conclusion

- concept of APC-DynaMOSA
- pre-selected configuration
- pre-trained classification model
- mixed results improved and degraded Java classes

# Future work

- long jump approach
- landscape feature
- real world use
- Fitness function for guidance

# Bibliography

Kauffman, S. and Levin, S. (1987).
*Towards a general theory of adaptive walks on rugged landscapes*, volume 128.


UNIVERSITÄT
PADERBORN