UNIVERSITÄT PADERBORN

Software Engineering

Landscape depending parameter tuning for search-based software testing

Overview

Introduction

- software tests are essential
- maximize coverage (line, branch, exception)
- lack of sufficient tests
- costly and time-consuming
- using search-based software testing

Motivation

- many tools solve this problem
- EvoSuite one of the bests in competitions [Panichella et al., 2020]
  - state-of-the-art: DynaMOSA
  - fairly good, but not perfect [Arcuri and Briand, 2014]
- optimal only with optimal configuration
- No Free Lunch theorem [Wolpert and Macready, 1997]
  - impossible to find optimal configuration

Research goal

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

State-of-the-art

- DynaMOSA sensitive for adaptive parameter control [Wetzler, 2020]
- parameter control can have a negative impact [Paterson et al., 2015]
- random can out-perform genetic [Shamshiri and Rojas, 2015]
- landscape characteristics [Albunian, 2020]

Delimitation

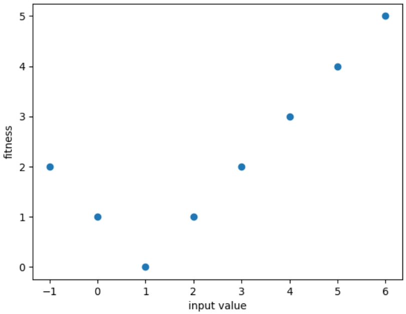
- focus on DynaMOSA
- pre-implemented parameter
- limit the experiment size
- use landscape features from the literature

Search-based software testing

- tests for object oriented languages
- sequence of method calls as search space
- maximizing coverage
- optimization problem
- using genetic algorithms

Fitness function

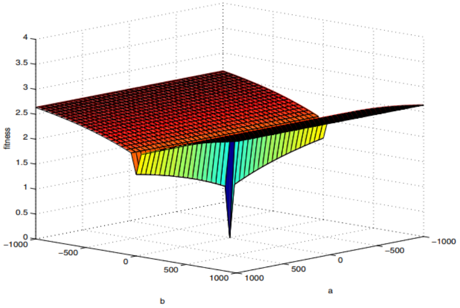
- function for e.g. branch distance
- guidance for search algorithms




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

Fitness landscape

- metaphor for the search space
- fitness value as height
- input as depth and width
- search for minima



Genetic algorithm

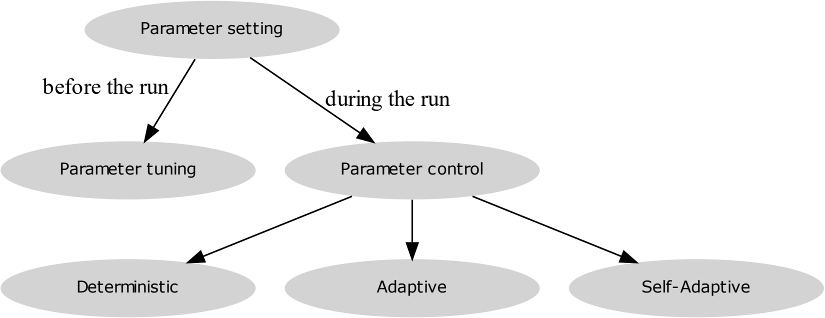
- start with random population
- iterate till termination condition
  - mutate and crossover
- return last generation

DynaMOSA

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

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

Parameter tuning and control



[Eiben et al., 1999]

Corpus

SF110

- 110 open-source Java projects
- 23,894 Java Classes
- trivial and non-trivial

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

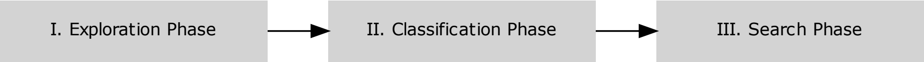
- S_3
- 20 Java Classes
- extracted from S_1
- high Standard Deviation
- 6.6 hours on tree machines

Comparisons

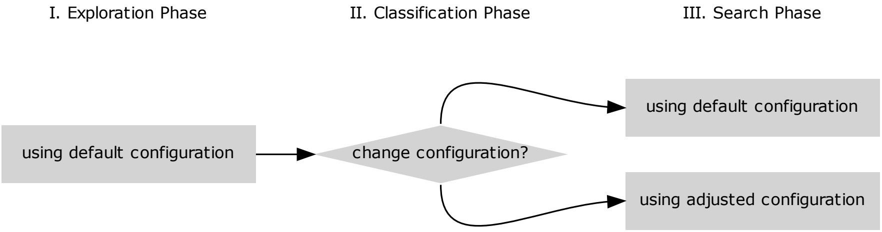
- 30 repeats for every Java Class
- Mann-Whitney U-test
- Vargha-Delaney effect size $\hat{A}_{12}$

Concept

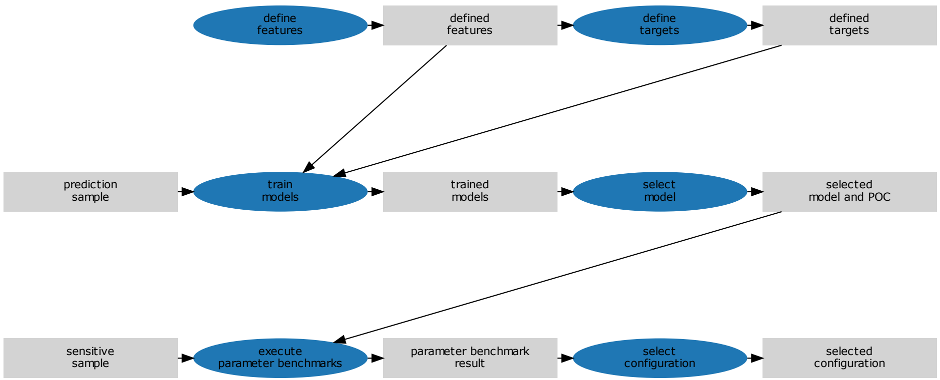
- use the beginning of the search to explore the search space
- classify the current search (”adjust” or ”default”)
- use configuration depending on class



Configuration selection

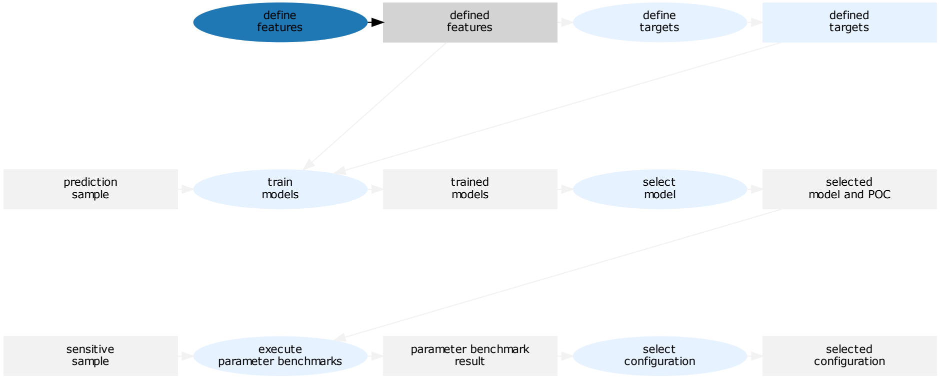


Components



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

Fitness

- ratio between:
  - current fitness value
  - maximum observed value
- range of $[0, 1]$
- high ratio: good performing
- low ratio: bad performing

Gradient branches

- [Shamshiri and Rojas, 2015]
- Byte-code analysis
- branch distance to be covered
- some branches without distance
- ratio between:
  - branches with gradient
  - branches without gradient
- range of $[0, 1]$
- high ratio: guidance
- low ratio: no guidance

Neutrality Volume

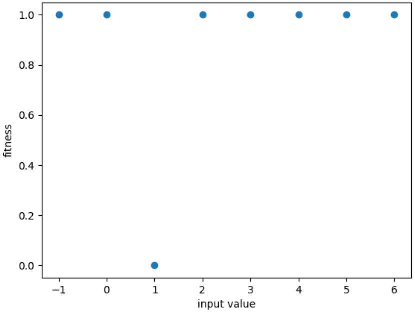
- [Albunian, 2020]
- neutral: no guidance
- based on fitness sequence S
- NV
  - fitness changes in S
  - depending on steps
- NV_{Ratio}:
  - range of $[0, 1]$

Example: Neutrality Volume

$$S = \{1, 1, 0, 1, 1, 1, 1, 1\} \qquad (1)$$

$$NV = 3 \qquad (2)$$

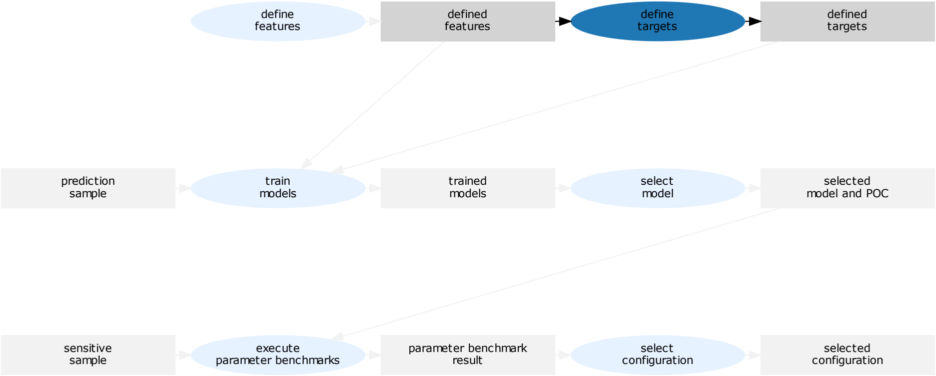
$$NV_{Ratio} = \frac{NV}{\#ofsteps} = \frac{3}{8} = 0.375 \qquad (3)$$



Information Content

- [Vassilev et al., 2000]
- how many information is needed to construct S

Classification target



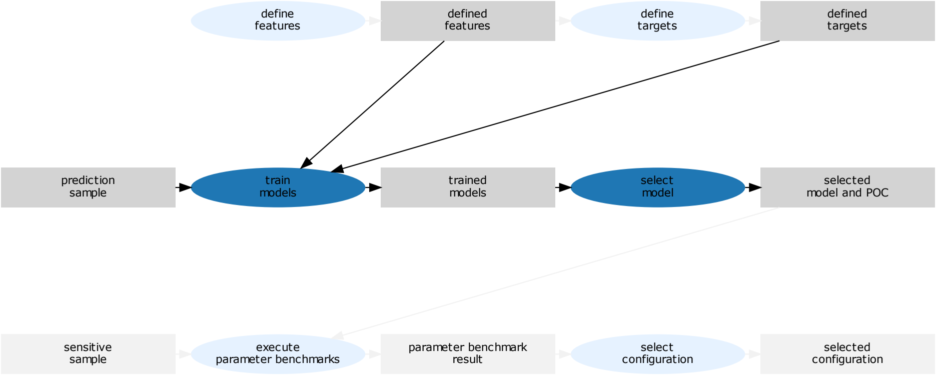
Targets

- Coverage improvement by parameter
  - $pop = 125$
- Low end coverage
  - $endcov. < 0.8$
- High standard deviation
  - $stdev > 0.1$
  - $stdev > 0$
- Relative low coverage
  - $cov. < max(cov.) * 0.8$

Targets

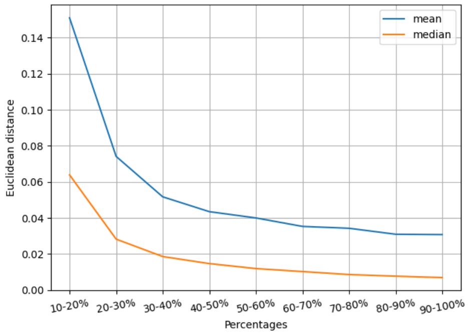
target	n in class	description
$cov. < max(cov.) * 0.8$	447	End coverage less than 80% of the best execution
$endcov. < 0.8$	6687	End coverage less than 80%
$stdev > 0.1$	541	Standard deviation greater than 0.1
$stdev > 0$	6046	Standard deviation greater than 0.0
$p = 125$	1735	The median coverage with 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



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

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

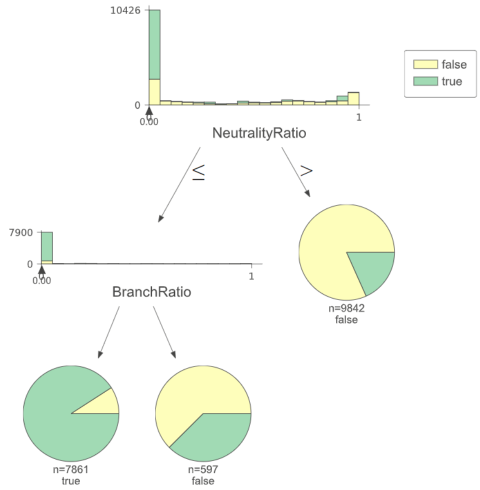
Hyper-parameter search

- 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

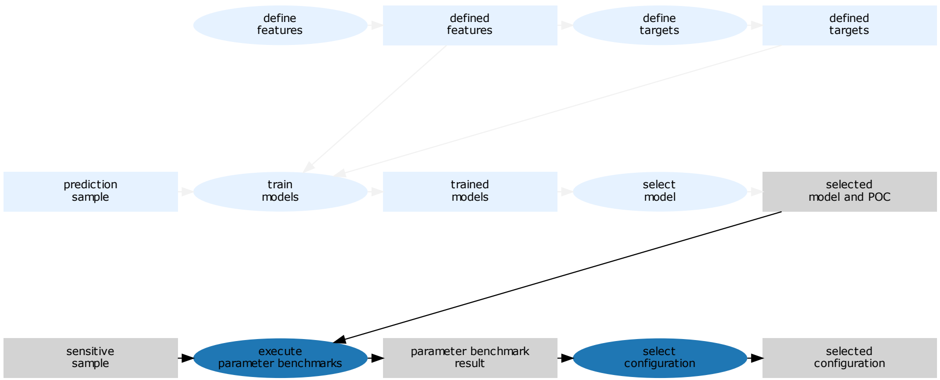
Component selection

Decision tree

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



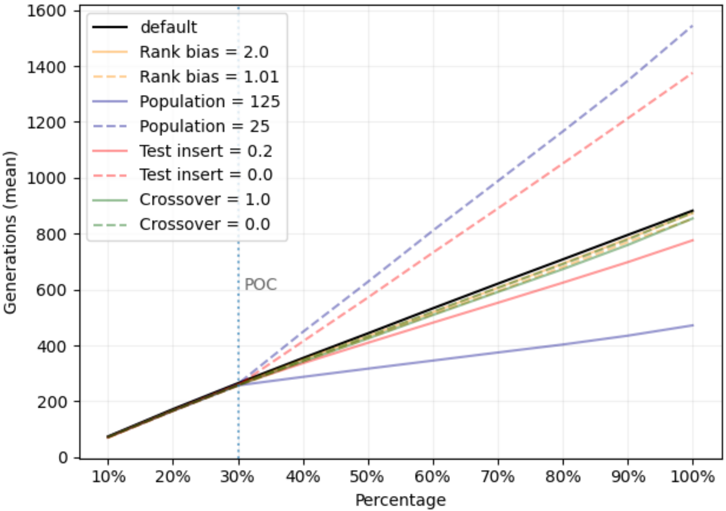
Parameter selection



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

$$P_{adj}(e) = \frac{n_1 P_1(e) + n_2 P_2(e)}{n_1 + n_2} \tag{4}$$

$$\hat{P}(e) = \sum_{i=k}^{n} \frac{n!}{i!(n-i)!} P_{adj}(e)^i (1 - P_{adj}(e))^{n-i} \tag{5}$$

Component selection

			P_{adj}			$\hat{P}$ for $k = 5$				
configuration	gen.	n	mut.	cross.	ins.	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**

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

CUT	coverage DynaMOSA mean	std	coverage APC-DynaMOSA mean	std	$\hat{A}_{12}$	p-value	
o.d.j.m.AbstractDBObject	**0.951**	**0.018**	**0.960**	**0.039**	**0.880**	**0.000**	↗
d.o.f.g.s.RoundStatsDiagram	**0.505**	**0.466**	**0.000**	**0.000**	**0.935**	**0.000**	↘
c.a.a.c.d.t.u.i.p.DHTUDPPacketHandlerStats	**0.909**	**0.010**	**0.894**	**0.020**	**0.917**	**0.001**	↘
o.p.g.d.VisualPageListItem	**0.112**	**0.001**	**0.113**	**0.002**	**0.933**	**0.003**	↗
n.s.s.c.s.a.RollbackAction	**0.170**	**0.320**	**0.000**	**0.000**	**0.935**	**0.006**	↘
f.v.n.m.b.s.p.w.p.GetParametersForm	**0.111**	**0.152**	**0.219**	**0.146**	**0.935**	**0.008**	↗
d.h.l.e.i.e.AbstractMessageBasedEventProducer	**0.426**	**0.018**	**0.439**	**0.023**	**0.913**	**0.020**	↗
c.i.s.LocalFileBrowser	**0.126**	**0.128**	**0.194**	**0.109**	**0.902**	**0.034**	↗
o.j.j.a.c.a.IndexedAceFileDataStore	**0.196**	**0.002**	**0.195**	**0.000**	**0.902**	**0.042**	↘
n.s.s.f.g.ErrorDialog	**0.021**	**0.073**	**0.126**	**0.138**	**0.611**	**0.046**	↗

Adaptive parameter control

Unknown sample S_2

CUT	coverage DynaMOSA mean	coverage DynaMOSA std	coverage APC-DynaMOSA mean	coverage APC-DynaMOSA std	$\hat{A}_{12}$	p-value	
c.g.f.FtpApplet	**0.065**	**0.066**	**0.099**	**0.056**	**0.567**	**0.034**	↗
o.a.c.m.d.DfpDec	0.013	0.038	0.034	0.057	0.278	0.077	
c.g.j.r.j.RecordType	0.835	0.019	0.834	0.012	0.278	0.126	
o.o.s.a.u.UpdateUserPanel	0.009	0.051	0.037	0.097	0.588	0.169	
n.s.j.m.a.q.TemplateUserTitles	0.055	0.064	0.077	0.064	0.750	0.203	
c.g.c.b.Predicates	0.510	0.039	0.519	0.037	0.317	0.414	
c.e.s.j.Room3D	0.083	0.120	0.108	0.126	0.671	0.435	
t.TwitterBaseImpl	0.551	0.010	0.548	0.022	0.720	0.734	
g.a.GroupAgent	0.723	0.075	0.734	0.071	0.700	0.873	

Selected configuration

Unknown sample S_2

CUT	coverage default		coverage adjusted config.				
	mean	std	mean	std	$\hat{A}_{12}$	p-value	
n.s.s.m.x.TableMeta	0.880	0.048	0.291	0.346	0.902	0.000	↘
n.v.a.g.r.RobotRenderer	0.628	0.094	0.729	0.146	0.902	0.004	↗
m.s.SSHSCPGUIThread	0.418	0.133	0.527	0.133	0.902	0.000	↗
o.a.c.m.d.f.MultivariateNormalMixtureExpectationMaximization	0.527	0.028	0.684	0.026	0.902	0.000	↗

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

Albunian, N. (2020).
An Investigation of Search Behaviour in Search-Based Unit Test Generation.
PhD thesis.

Arcuri, A. and Briand, L. (2014).
A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering.
Software Testing, Verification and Reliability, 24(3):219–250.

Eiben, A. E., Hinterding, R., and Michalewicz, Z. (1999).
Parameter control in evolutionary algorithms.

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

Panichella, A., Campos, J., and Fraser, G. (2020).
Evosuite at the sbst 2020 tool competition.


UNIVERSITÄT
PADERBORN