

Software Engineering

Landscape depending parameter tuning for search-based software testing

by Kevin Haack

Thesis Supervisor: Dr. Thomas Vogel



Overview

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

Overview

1. Introduction

2. Fundamentals

3. Experimental setup

4. Adaptive parameter control

5. Evaluation

6. Conclusion

Introduction

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

Search-based software testing (SBST)

- automatically generating unit tests
- sequence of method calls as search space
- maximizing coverage
- treat as: optimization problem
- using genetic algorithms

Motivation

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

Research goal

- EvoSuite
 - good at: wide variety of problem-cases
 - bad at: some problem-cases
- concept landscape depending adaptive parameter control
- increase coverage of EvoSuite

State-of-the-art

- [Wetzler, 2020]: DynaMOSA sensitive for adaptive parameter control
- [Paterson et al., 2015]: parameter control can have a negative impact
- [Shamshiri and Rojas, 2015]: random can out-perform a genetic algorithm
- [Albunian, 2020]: landscape characteristics
 - Information Content (IC)
 - Neutrality Volume (NV)

Delimitation

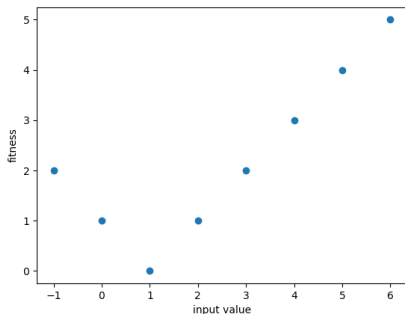
- focus on DynaMOSA
- only implemented parameter
- limit the experiment size
- use landscape features from the literature

Overview

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

Fitness function

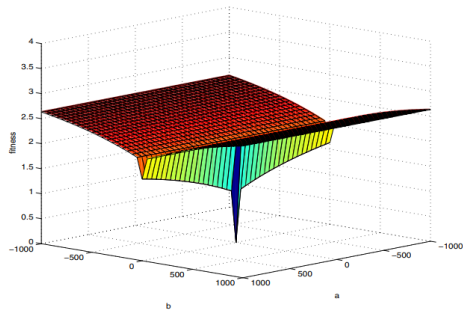
- function for e.g. branch distance
- guidance for search algorithms



```
1 void bar(int x) {  
2     if (x == 1) {  
3         // uncovered code  
4     }  
5 }
```

Fitness landscape

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

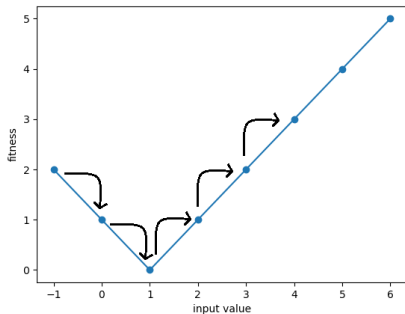


[Harman, 2007]

Random Walk

[Kauffman and Levin, 1987]

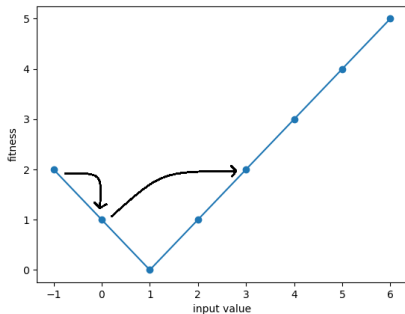
- used to describe landscape
- walk over landscape
- random unbiased steps



Long Jump

[Kauffman and Levin, 1987]

- multiple steps
- not precise
- skip features (like minima)
- landscape approximation



Genetic algorithm

- unit test = individual
- multiple tests = population
- start with random population
- iterate till termination condition
 - mutate and crossover
- return last generation

DynaMOSA

[Panichella et al., 2017]

- genetic algorithm
- multiple target
- dynamic target update
- archive: keep track of target covering
- return archive as last generation individuals

Overview

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

Experiments

- multiple experiments
- executed on 3 identical machines
- 30 repeats for every Java Class
- Mann-Whitney U-test
- Vargha-Delaney effect size \hat{A}_{12}

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

Samples

S_1

- for training prediction
- corpus: SF110
- randomly selected
- 709 Java Classes
- 9.8 days

S_2

- for evaluation
- corpus: Panichella
- all classes
- 346 Java Classes
- 4.8 days

S_3

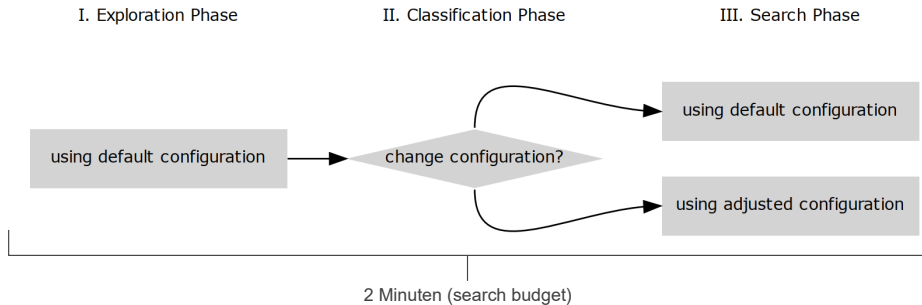
- for parameter search
- corpus: SF110
- high stdev
- 20 Java Classes
- 6.6 hours

Overview

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

Concept

- 2 minutes search budget
- 3 phases



Research questions

- RQ 1. landscape features
- RQ 2. classification target
- RQ 3. classification model
- RQ 4. percentage of classification (POC)
- RQ 5. adjusted configuration

Research questions

- RQ 1. landscape features
- RQ 2. classification target
- RQ 3. classification model
- RQ 4. percentage of classification (POC)
- RQ 5. adjusted configuration

Landscape approximation

[Kauffman and Levin, 1987]

- random walk is time-consuming
- instead: using the first generations
- "long jump" instead a step
- approximation of the landscape

Fitness

- implemented in EvoSuite
- ratio between:
 - current fitness value
 - maximum observed value
- high ratio: good performing
- low ratio: bad performing

Gradient branches

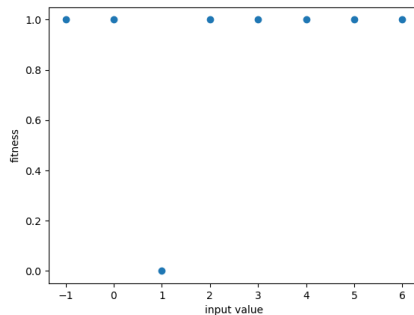
[Shamshiri and Rojas, 2015]

- implemented in EvoSuite
- Byte-code analysis
 - branches with gradient
 - branches without gradient
- ratio between both
- high ratio: guidance
- low ratio: no guidance

Neutrality Volume (NV)

[Albunian, 2020]

- neutrality of the landscape
- neutral: no guidance
- based on fitness values
- number of fitness changes
- depending on steps
 - NV_{Ratio}



Information Content (IC)

[Vassilev et al., 2000, Albunian, 2020]

- information content of the landscape
- ruggedness of the landscape
- high IC: rugged landscape
- low IC: soft landscape

Research questions

- RQ 1. landscape features
- RQ 2. classification target
- RQ 3. classification model
- RQ 4. percentage of classification (POC)
- RQ 5. adjusted configuration

Classification targets

- predict: coverage improvement by parameter
- problems
 - nearly infinite possible combinations
 - time-consuming experiments
 - example experiment shows: low TPR
- instead: use data of one experiment
- targets approximations

Targets approximations

- predict: low end coverage
 - search will not reach 100% or 80% coverage
- predict: relative low coverage
 - search will not reach 100% or 80% of the highest coverage for this Java Class
- predict: high standard deviation
 - coverage of the Java Class has high stdev

Research questions

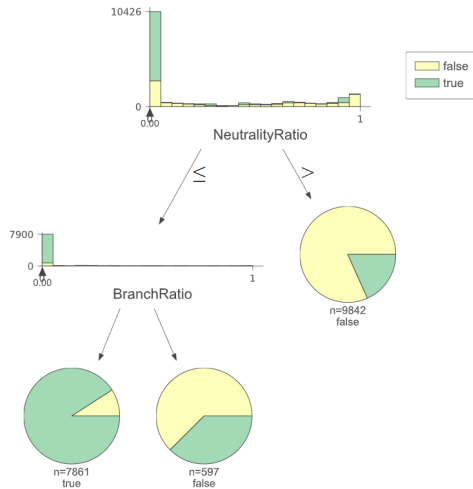
- RQ 1. landscape features
- RQ 2. classification target
- RQ 3. classification model
- RQ 4. percentage of classification (POC)
- RQ 5. adjusted configuration

Classification model

- using sample S_1
- generate multiple decision trees
 - hyper-parameter search
 - every POC
 - every target
- select the best performing

Decision tree

- target: high stdev and relative low coverage
- depth of decision tree: 2
- Gini Impurity
- POC: 30%



Research questions

- RQ 1. landscape features
- RQ 2. classification target
- RQ 3. classification model
- RQ 4. percentage of classification (POC)
- RQ 5. adjusted configuration

Configuration selection

- using sample S_3
- many EvoSuite parameter
 - nearly infinite possible combinations
 - influence on runtime
 - influence on probability of mutation/crossover/insert
- small-scale experiments

Configuration experiments

configuration	n_{+}	n_{-}
$p = 25$	0	0
$p = 125$	3	0
$cr = 0.0$	1	0
$cr = 1.0$	0	0
$bias = 2.0$	1	0
$bias = 1.01$	0	0
$pti = 0.0$	0	4
$pti = 0.2$	0	0
$p = 10, pti = 5.0$	3	0
$p = 25, pti = 1.0$	4	0

Overview

1. Introduction
2. Fundamentals
3. Experimental setup
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%

- RQ1 landscape feature ✓
- RQ2 classification target
- RQ3 classification model ✓
- RQ4 POC ✓
- RQ5 configuration

Classification

Unknown sample S_2

- trained on S_1
- test on S_2
- TPR of 21%
- FPR of 7%

- RQ1 landscape feature χ
- RQ2 classification target
- RQ3 classification model χ
- RQ4 POC χ
- RQ5 configuration

Adaptive parameter control

Known sample S_1

- 709 Java Classes
- 50 Java Classes "adjusted"
- 10 significant change
 - 6 positive change (up to 10.8%)
 - 4 negative change (up to 50.5%)
- mixed result

- RQ1 landscape feature ✓
- RQ2 classification target ✗
- RQ3 classification model ✓
- RQ4 POC ✓
- RQ5 configuration ✗

Adaptive parameter control

Unknown sample S_2

- 346 Java Classes
- 22 Java Classes "adjusted"
- 1 significant change
 - 1 positive change (with 3.4%)
 - 0 negative change
- positive result

- RQ1 landscape feature ✗
- RQ2 classification target ✓
- RQ3 classification model ✗
- RQ4 POC ✗
- RQ5 configuration ✓

Configuration separately

Unknown sample S_2

- without the concept
- 346 Java Classes
- 95 significant change
 - 58 positive change
 - 37 negative change
- mixed result
 - concept avoid the negative impact

- RQ1 landscape feature
- RQ2 classification target
- RQ3 classification model ✓
- RQ4 POC
- RQ5 configuration ✗

Overview

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





Conclusion

- concept of APC-DynaMOSA
- select configuration
- trained classification model
- mixed results
 - reliable prediction on S_1
 - no suitable configuration




Future work

- "long jump" approach
 - is it suitable during the first generations
- landscape feature
- real world use of SBST
 - support of new JDK
 - support of new IDE
 - support of new frameworks
- fitness function for guidance





Bibliography I

-  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.
-  Harman, M. (2007).
The current state and future of search based software engineering.
In *Future of Software Engineering (FOSE '07)*. IEEE.

Bibliography II

-  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.
In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, ACM Digital Library, pages 549–552, New York, NY, United States. Association for Computing Machinery.
-  Panichella, A., Kifetew, F., and Tonella, P. (2017).
Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets.
IEEE Transactions on Software Engineering, pages 122–158.

Bibliography III

-  Paterson, D., Turner, J., White, T., and Fraser, G. (2015).
Parameter control in search-based generation of unit test suites.
pages 141–156. Springer, Cham.
-  Shamshiri, S. and Rojas, J. M. (2015).
Random or genetic algorithm search for object-oriented test suite generation?
-  Vassilev, V. K., Fogarty, T. C., and Miller, J. F. (2000).
Information characteristics and the structure of landscapes.
Evolutionary computation, 8(1):31–60.
-  Wetzler, K. (2020).
Analyzing parameter control in search-based test case generation techniques.

Bibliography IV



Wolpert, D. H. and Macready, W. G. (1997).

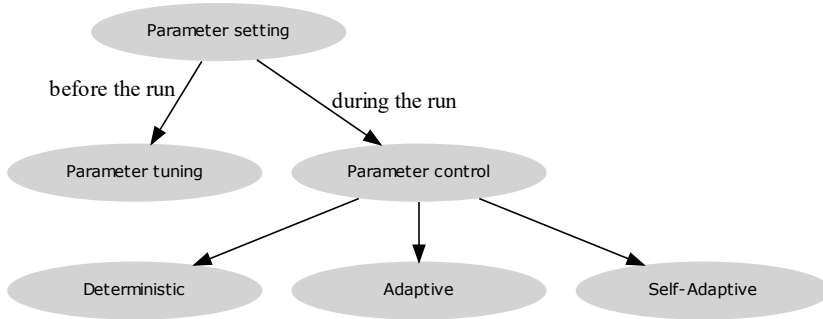
No free lunch theorems for optimization.

IEEE Transactions on Evolutionary Computation, pages 67–82.



UNIVERSITÄT
PADERBORN

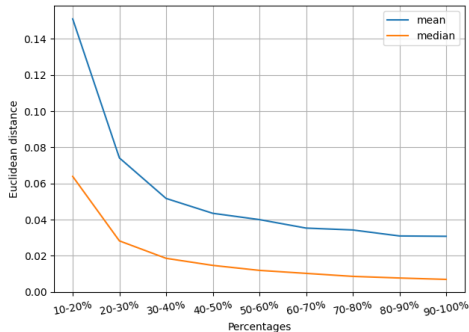
Parameter tuning and control



[Eiben et al., 1999]

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)



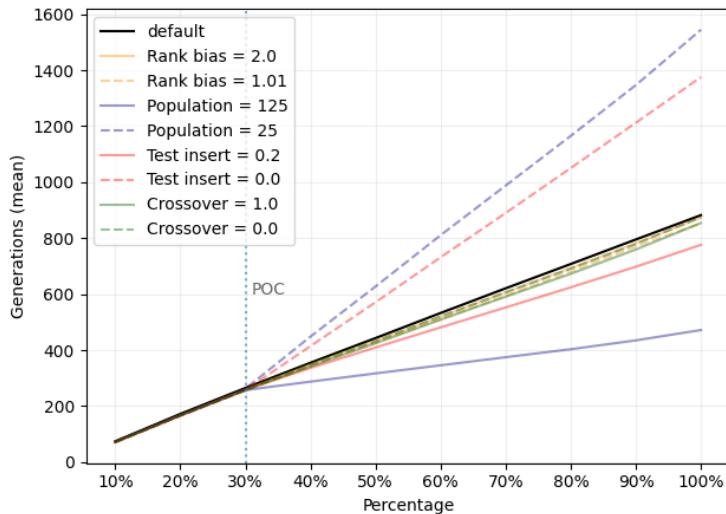
Machine learning algorithm

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

Hyper-parameter search

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

Influence on runtime



Event probability

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

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

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

Parameter experiments

configuration	gen.	n	P_{adj}			\hat{P} for $k = 5$			n_{+}	n_{-}
			mut.	cross.	ins.	mut.	cross.	ins.		
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 ↗	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 ↗	0.0	0	0
$bias = 2.0$	912	9	0.95	0.68	0.05	1.0	0.88 ↗	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 ↗	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 ↗	3	0
$p = 25, pti = 1.0$	777	8	0.75	0.38	0.25	0.89	0.14	0.03 ↗	4	0

Adaptive parameter control

Known sample S_1

CUT	coverage DynaMOSA		coverage APC-DynaMOSA		\hat{A}_{12}	p-value	
	mean	std	mean	std			
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 I

Unknown sample S_2

CUT	coverage DynaMOSA		coverage APC-DynaMOSA		\hat{A}_{12}	p-value
	mean	std	mean	std		
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.TwitterBaselImpl	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.		\hat{A}_{12}	p-value	
	mean	std	mean	std			
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	↗