Basic Block Coverage for Unit Test Generation at the SBST 2022 Tool Competition

Derakhshanfar, Pouria; Devroey, Xavier

DOI
10.1145/3526072.3527528

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the 2022 IEEE/ACM 15th International Workshop on Search-Based Software Testing (SBST)

Basic Block Coverage for Unit Test Generation at the SBST 2022 Tool Competition

Pouria Derakhshanfar
Delft University of Technology
Delft, Netherlands
p.derakhshanfar@tudelft.nl

Xavier Devroey
NADI, University of Namur
Namur, Belgium
xavier.devroey@unamur.be

ABSTRACT

Basic Block Coverage (BBC) is a secondary objective for search-based unit test generation techniques relying on the approach level and branch distance to drive the search process. Unlike the approach level and branch distance, which considers only information related to the coverage of explicit branches coming from conditional and loop statements, BBC also takes into account implicit branchings (e.g., a runtime exception thrown in a branchless method) denoted by the coverage level of relevant basic blocks in a control flow graph to drive the search process. Our implementation of BBC for unit test generation relies on the DynaMOSA algorithm and EvoSuite. This paper summarizes the results achieved by EvoSuite's DynaMOSA implementation with BBC as a secondary objective at the SBST 2022 unit testing tool competition.

CCS CONCEPTS

• **Software and its engineering** → **Search-based software engineering**; **Software testing and debugging**.

KEYWORDS

basic block coverage, search-based unit test generation, EvoSuite

ACM Reference Format:
Pouria Derakhshanfar and Xavier Devroey. 2022. Basic Block Coverage for Unit Test Generation at the SBST 2022 Tool Competition. In *The 15th Search-Based Software Testing Workshop (SBST'22), May 9, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3526072.3527528

1 INTRODUCTION

Various techniques have been developed over the years to generate unit tests for Java programs. For the 10th time, the Java Unit Testing Tool Competition, co-located with the 15th edition of the International Workshop on Search-Based Software Testing (SBST 2022), has evaluated several unit test generators on an unknown set of benchmarks to compare the generated tests in terms of structural coverage and mutation score [4, 6]. In this short paper, we report the results of our implementation of Basic Block Coverage (BBC), a secondary objective for search-based test case generation [2, 3], on top of EvoSuite [5], a state-of-the-art unit test generator for Java, at the SBST 2022 Tool Competition.

SBST'22, May 9, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9318-8/22/05.
https://doi.org/10.1145/3526072.3527528

2 BASIC BLOCK COVERAGE

BBC was originally proposed for search-based crash reproduction [1, 2]. Its main purpose is to account for partial coverage caused by implicit branchings in basic blocks. For instance, many basic Java operations (e.g., access to an array, method call on an object, etc.) can throw runtime exceptions (e.g., `ArrayIndexOutOfBoundsException`, `NullPointerException`, etc.) that are not declared in the header of the encapsulating method. When thrown, such exceptions cause implicit branching in the program that are ignored by search-based unit test generation techniques relying on the approach level and branch distance heuristics to drive the search process. Indeed, both heuristics hypothesize that only a limited number of basic blocks can change the execution path away from a target statement. However, if, for instance, a runtime exception is thrown in the middle of a statement, then the search process does not benefit from any further guidance from the approach level and branch distance to reach the targeted statement.

More recently, we extended and evaluated the application of BBC to search-based unit test generation with DynaMOSA [3, 7].[1] Unlike crash reproduction, where only a limited number of paths need to be explored to reach a target statement and reproduce a crash, unit test generation seeks to cover several different target statements, depending on the objectives defined for the search. In both cases, BBC is used as a secondary objective to compare two test cases with the same distance to a target statement according to the approach level and branch distance to find out which one is the closest. We implemented BBC in EvoSuite [5]

Parameter settings. We rely on EvoSuite with the DynaMOSA algorithm, and the default set of coverage criteria enabled [8] (i.e., line coverage, branch coverage, branch coverage by direct method invocations, weak mutation testing, output coverage, exception coverage). We set the secondary objectives to maximize BBC and minimize the length to avoid an explosion of the size of the tests during the search (`-Dsecondary_objectives=BBCOVERAGE:TOTAL_LENGTH`). BBC comes with two additional parameters: the *usage rate*, defining the probability of activating BBC when two test cases reach the same distance for a given target, and the *sleep time*, defining the delay after which BBC can be activated when DynaMOSA adds a target to the active search objectives. We left the usage rate to its default value of 0.5 (`-DBBC_USAGE_PERCENTAGE=50`), as recommended from our previous evaluation [3]. We used a sleep time of 10 seconds (`-DBBC_SLEEP_TIME=10`) when the total search budget is less than 60 seconds, 30 seconds when the total search budget is less than 300 seconds, and 60 seconds otherwise. The competition

[1] Our implementation is openly available at https://github.com/pderakhshanfar/evosuite/tree/BBC.

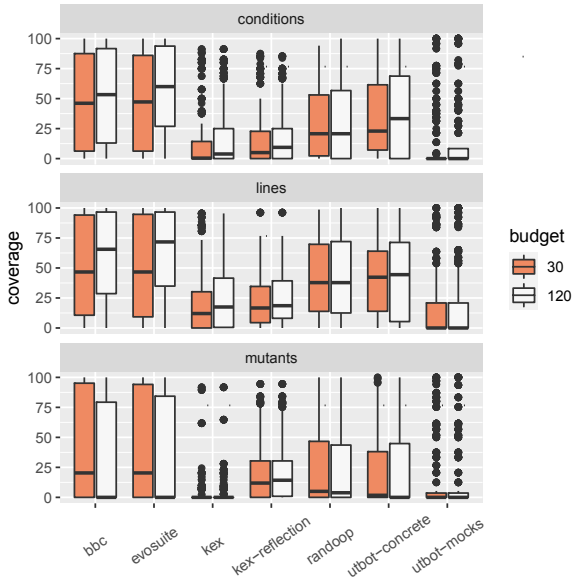

Figure 1: Overall Results of the SBST 2022 Java Unit Testing Tool Competition.

Table 1: Number of benchmarks for which BBC performed better ($\hat{A}_{12} < 0.5$) or worse ($\hat{A}_{12} > 0.5$) than EvoSuite with a medium or large magnitude and a significance level of 0.01.

	30 sec.		120 sec.	
	< 0.5	> 0.5	< 0.5	> 0.5
lines coverage	**2**	**10**	**5**	**10**
medium	1	6	3	7
large	1	4	2	3
conditions coverage	**1**	**6**	**4**	**9**
medium	1	1	2	4
large	0	5	2	5
mutation score	**2**	**4**	**3**	**6**
medium	2	3	2	3
large	0	1	1	3

ran with two different search budgets this year: 30 and 120 seconds. All the other parameters were left to their default value.

3 RESULTS

Figure 1 reports the overall coverages for conditions and lines and the mutation score of the different tools entering the 2022 competition. In general, the lines and conditions coverage increases for BBC when increasing the time budget. However, we see a decrease in the mutation score. This decrease should be further investigated by looking at the generated tests. One possible explanation could be the difference in the number of generated tests: 18,963 tests in total across the different runs for a 30 seconds time budget against 15,561 tests in total across the different runs for a 120 seconds budget.

As expected, the results of BBC are close to the ones of EvoSuite. This is in line with our previous evaluation of BBC for unit test generation [3]. There is, however, a difference in the coverages and mutation scores of the different benchmarks between EvoSuite and BBC. This difference is confirmed by our analysis using the non-parametric Wilcoxon Rank Sum test (with $\alpha = 0.01$) and effect size Vargha-Delaney $\hat{A}_{12}$ statistic, reported in Table 1. As can be seen from the Table, BBC and EvoSuite seem to cover the benchmarks differently, and this difference seems to evolve over time. Further investigation is needed to identify the factors influencing the effectiveness of BBC for specific benchmarks (like, for instance, the presence of implicit branches, sleep time for small search budgets, etc.).

4 CONCLUSION

This short paper presents the results of Basic Block Coverage for unit test generation at the 10th Java Unit Testing Tool Competition. BBC achieved a good score, ranking second out of seven tools taking part in the competition this year. The results show that BBC covers the different benchmarks differently, compared to EvoSuite. Further investigations are needed to identify the best conditions for BBC to operate, especially with small budgets.

ACKNOWLEDGMENTS

We would like to thank the organizers of the 10th Java Unit Testing Tool Competition. This research was partially funded by the EU Horizon 2020 *COSMOS* (DevOps for Complex Cyber-physical Systems) Project No. 957254-COSMOS, and the CyberExcellence (No. 2110186) project, funded by the Public Service of Wallonia (SPW Recherche).

REFERENCES

[1] Pouria Derakhshanfar, Xavier Devroey, Annibale Panichella, Andy Zaidman, and Arie Van Deursen. 2020. Botsing, a Search-based Crash Reproduction Framework for Java. In *35th IEEE/ACM International Conference on Automated Software Engineering (ASE '20), September 21–25, 2020, Virtual Event, Australia.* ACM/IEEE, 1278–1282. https://doi.org/10.1145/3324884.3415299

[2] Pouria Derakhshanfar, Xavier Devroey, and Andy Zaidman. 2020. It Is Not Only About Control Dependent Nodes: Basic Block Coverage for Search-Based Crash Reproduction. In *Search-Based Software Engineering - 12th International Symposium, SSBSE 2020,* Aldeida Aleti and Annibale Panichella (Eds.). Springer, 42–57. https://doi.org/10.1007/978-3-030-59762-7_4

[3] Pouria Derakhshanfar, Xavier Devroey, and Andy Zaidman. 2022. Basic Block Coverage for Search-based Unit Testing and Crash Reproduction. https://doi.org/10.48550/arXiv.2203.02337 arXiv:2203.02337 [cs.SE]

[4] Xavier Devroey, Alessio Gambi, Juan Pablo Galeotti, René Just, Fitsum Kifetew, Annibale Panichella, and Sebastiano Panichella. 2021. JUGE: An Infrastructure for Benchmarking Java Unit Test Generators. https://doi.org/10.48550/arXiv.2106.07520 arXiv:2106.07520 [cs.SE]

[5] Gordon Fraser and Andrea Arcuri. 2011. EvoSuite: Automatic Test Suite Generation for Object-Oriented Software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering - SIGSOFT/FSE '11 (ESEC/FSE '11).* ACM Press, 416. https://doi.org/10.1145/2025113.2025179

[6] Alessio Gambi, Gunel Jahangirova, Vincenzo Riccio, and Fiorella Zampetti. 2022. SBST Tool Competition 2022. In *15th IEEE/ACM International Workshop on Search-Based Software Testing, SBST 2022.* IEEE/ACM, Pittsburgh, PA, USA.

[7] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2018. Automated Test Case Generation as a Many-Objective Optimisation Problem with Dynamic Selection of the Targets. *IEEE Transactions on Software Engineering* 44, 2 (Feb. 2018), 122–158. https://doi.org/10.1109/TSE.2017.2663435

[8] José Miguel Rojas, José Campos, Mattia Vivanti, Gordon Fraser, and Andrea Arcuri. 2015. Combining Multiple Coverage Criteria in Search-Based Unit Test Generation. In *Search-Based Software Engineering (SSBSE 2015) (LNCS, Vol. 9275).* 93–108. https://doi.org/10.1007/978-3-319-22183-0_7