Crash Reproduction Using Helper Objectives

Derakhshanfar, Pouria; Devroey, Xavier; Zaidman, Andy; van Deursen, Arie; Panichella, Annibale

DOI
10.1145/3377929.3390077

Publication date
2020

Document Version
Accepted author manuscript

Published in
GECCO 2020 Companion - Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion

Crash Reproduction Using Helper Objectives

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

Xavier Devroey
x.d.m.devroey@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Andy Zaidman
a.e.zaidman@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Arie van Deursen
arie.vandeursen@tudelft.nl
Delft University of Technology
Delft, The Netherlands

Annibale Panichella
a.panichella@tudelft.nl
Delft University of Technology
Delft, The Netherlands

ABSTRACT

Evolutionary-based crash reproduction techniques aid developers in their debugging practices by generating a test case that reproduces a crash given its stack trace. In these techniques, the search process is typically guided by a single search objective called *Crash Distance*. Previous studies have shown that current approaches could only reproduce a limited number of crashes due to a lack of diversity in the population during the search. In this study, we address this issue by applying Multi-Objectivization using Helper-Objectives (*MO-HO*) on crash reproduction. In particular, we add two helper-objectives to the *Crash Distance* to improve the diversity of the generated test cases and consequently enhance the guidance of the search process. We assessed *MO-HO* against the single-objective crash reproduction. Our results show that *MO-HO* can reproduce two additional crashes that were not previously reproducible by the single-objective approach.

CCS CONCEPTS

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

KEYWORDS

crash reproduction, search-based software testing, MOEA

ACM Reference Format:
Pouria Derakhshanfar, Xavier Devroey, Andy Zaidman, Arie van Deursen, and Annibale Panichella. 2020. Crash Reproduction Using Helper Objectives. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3390077

1 INTRODUCTION

When a software application crashes, a report (or issue), including information gathered during the crash, is assigned to developers for debugging. One common practice to identify the root cause of a crash is to write a test case reproducing it [8]. This test case can later be adapted and integrated into the test suite to prevent future regressions. However, depending on the amount of information available in the report, writing this *crash reproducing test case* can be time-consuming and labor-intensive [7].

Consequently, various approaches in the literature try to automate *crash reproduction*. The earlier empirical study [7] shows that search-based crash reproduction outperforms other crash reproduction approaches in terms of *crash reproduction ratio* (percentage of crashes that could be reproduced) and *efficiency* (time taken to reproduce a given crash successfully). Search-based crash reproduction generates a test case that, when executed, is able to reproduce that crash by modeling the crash reproduction problem as an optimization problem. This approach reformulates crash reproduction as a single search objective (*Crash Distance* hereafter), which measures how far a generated test is from reproducing the crash, and applies a single-objective evolutionary algorithm (Single-Objective Search hereafter) to generate solutions (*i.e.,* tests).

Although Single-Objective Search performs well compared to other crash reproduction approaches, a more extensive empirical study [6] revealed that it is not successful in reproducing complex crashes (*i.e.,* large stack traces). Hence, further studies to enhance the guidance of the search process are required.

In this study, we investigate a new strategy to Multi-Objectivize crash reproduction based on Helper-Objectives (*MO-HO*) [3]. More specifically, we add two additional helper-objectives to *Crash Distance* (first objective): *method call diversity* (second objective) and *test case length minimization* (third objective). For the second objective, we re-use a distance function that measures the diversity of the methods called in the test cases. For the third objective, we count the number of statements in the generated test case. Since these three objectives conflit with *Crash Distance*, we expect an increase in the solutions' diversity and, hence, an improvement in crash reproduction effectiveness (crash reproduction ratio) and efficiency. We utilize SPEA2 [9], which is a multi-objective evolutionary algorithm (MOEA), to solve this optimization problem.

We compare our approach against Single-Objective Search [7] from the perspectives of *crash reproduction*. Our results show that *MO-HO* can reproduce new crashes, which are not reproducible with Single-Objective Search.

2 MULTI-OBJECTIVIZATION WITH HELPER-OBJECTIVES (MO-HO)

As suggested by Jensen *et al.* [3], adding helper-objectives, which are in conflict with the primary one, to an existing single objective can help search algorithms escape from local optima. Therefore,

GECCO '20 Companion, July 8–12, 2020, Cancún, Mexico

defining proper helper-objectives is crucial. In this study, we add two helper-objectives called **method sequence diversity** and **test length minimization** that aim to increase the diversity in the population (e.g., generated tests) and address the *bloating* effect. Then, we use SPEA2 MOEA to solve this optimization problem.

2.1 Method Sequence Diversity (MSD).

The first helper-objective seeks to maximize the diversity of the method-call sequences that compose the generated tests. To measure the value of this objective, we measure the diversity between two test cases by using a binary encoding scheme and calculating the distance between the corresponding encoded vectors using the Hamming distance [2]. For three or more test cases, the overall diversity corresponds to the average pairwise diversity of the existing test cases [4].

Let us assume that $F = \{f_1, f_2, ..f_n\}$ is a set of public and protected methods in the target class, and $T = \{t_1, t_2, ..t_m\}$ is a set of generated test cases. To calculate the diversity of T, we first need to encode each $t_k \in T$ to a binary vector. We use the same encoding scheme proposed by Mondal *et al.* [4]: each test case $t_k \in T$ corresponds to a binary vector v_k of length n. Each element $v_k[i]$ of the binary vector denotes whether the corresponding method $f_i \in F$ is invoked by the test case t_k. More formally, for each method $f_i \in F$, the corresponding entry $v_k[i] = 1$ if t_k calls f_i; $v_k[i] = 0$ otherwise. Then, we calculate the diversity for each pair of test cases t_k and t_j as the Hamming distance between the corresponding binary vectors v_k and v_j [2].

2.2 Test Length Minimization

If we apply *method sequence diversity* to our search problem, we may end up with long test cases [1]. Let us assume that we have a set of short test cases with few method calls in our population (most of the elements in their binary vectors are 0). A lengthy test case t_L that calls all the methods of the target class will have a binary vector of ones. As a consequence, t_L will have a large Hamming distance from the existing test cases. To avoid the *bloating* effect [5], our second helper-objective is *test length minimization*, seeking to minimize the number of statements in a given test.

3 EXPERIMENTAL ANALYSIS

We used 30 randomly selected crashes from JCrashPack [6], a collection of crashes for crash reproduction benchmarking. We attempted to reproduce the selected crashes using both *MO-HO* and Single-Objective Search with the search budget of five minutes and the population size of 50. In SPEA2, we set the archive size to 50. For both algorithms, we use the same genetic operators with the same parameter values. More precisely, fittest individuals (tests) are selected for reproduction using the *binary tournament selection*. New tests are generated using the *uniform mutation* with mutation probability $p_m = 1/n$ (where n is the length of the test case) and the *single-point crossover* with probability $p_c = 0.8$.

We examined if *MO-HO* can reproduce any additional crashes, which are not reproducible by Single-Objective Search. We observed that our approach could reproduce two new crashes: XWIKI-14227 and LANG-19b. Moreover, we observed that all of the crashes reproduced by Single-Objective Search could be reproduced by *MO-HO* as well.

4 CONCLUSION AND FUTURE WORK

Generating crash reproducing test cases can ease the process of debugging for developers. A promising approach for automating this process is using evolutionary algorithms. This approach defines an optimization objective called *Crash Distance* and applies a single objective guided evolutionary algorithm (Single-Objective Search). This strategy may end up generating test cases that are not diverse enough because of a low exploration during the search process.

In this initial study, we apply *MO-HO* to tackle the problem of the former technique. In *MO-HO*, we define two helper-objectives in addition to *Crash Distance* to alleviate the lack of exploration. Moreover, the introduced helper-objectives conflict with the main objective *Crash Distance*.

We assessed the application of *MO-HO* to SPEA2 (a commonly used MOEA) to solve the crash reproduction problem and compared its results against Single-Objective Search. Results indicate that *MO-HO* can reproduce two new crashes (6% of selected crashes) not reproduced by Single-Objective Search. Since our early results are encouraging, we seek to perform an empirical study (on more crashes) and characterize the contributing factors in *MO-HO* in a future study.

ACKNOWLEDGMENTS

This research was partially funded by the EU Horizon 2020 ICT-10-2016-RIA "STAMP" project (No.731529).

REFERENCES

[1] Nasser M Albunian. 2017. Diversity in search-based unit test suite generation. In *International Symposium on Search Based Software Engineering*. Springer, 183–189.
[2] R. W. Hamming. 1950. Error Detecting and Error Correcting Codes. *Bell System Technical Journal* 29, 2 (apr 1950), 147–160.
[3] Mikkel T Jensen. 2004. Helper-objectives: Using multi-objective evolutionary algorithms for single-objective optimisation. *Journal of Mathematical Modelling and Algorithms* 3, 4 (2004), 323–347.
[4] Debajyoti Mondal, Hadi Hemmati, and Stephane Durocher. 2015. Exploring test suite diversification and code coverage in multi-objective test case selection. In *2015 IEEE 8th International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 1–10.
[5] Annibale Panichella, Fitsum Meshesha Kifetew, and Paolo Tonella. 2017. Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Transactions on Software Engineering* 44, 2 (2017), 122–158.
[6] Mozhan Soltani, Pouria Derakhshanfar, Xavier Devroey, and Arie van Deursen. 2020. A benchmark-based evaluation of search-based crash reproduction. *Empirical Software Engineering* 25, 1 (jan 2020), 96–138.
[7] Mozhan Soltani, Annibale Panichella, and Arie Van Deursen. 2018. Search-Based Crash Reproduction and Its Impact on Debugging. *IEEE Transactions on Software Engineering* (2018), 1–1.
[8] Andreas Zeller. 2009. *Why Programs Fail, Second Edition: A Guide to Systematic Debugging* (2nd ed.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
[9] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. *TIK-report* 103 (2001).