M. Tichy, E. Bodden, M. Kuhrmann, S. Wagner, J.-P. Steghöfer (Hrsg.): SE 2018,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2018 75

Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration

Helge Spieker[1] Arnaud Gotlieb[1] Dusica Marijan[1] Morten Mossige[2]

Abstract: The paper appeared at the International Symposium on Software Testing and Analysis (ISSTA 2017) [Sp17]. It is part of a project on test case prioritization, selection, and execution in Continuous Integration (CI) (see also [Mo17]). Selecting the most promising test cases to detect bugs is hard if there are uncertainties on the impact of committed code changes or if traceability links between code and tests are not available. This paper introduces RETECS, a new method for automatically learning test case selection and prioritization in CI with the goal to minimize the round-trip time between code commits and developer feedback on failed test cases. RETECS uses reinforcement learning to select and prioritize test cases according to their duration, previous last execution and failure history. In a constantly changing environment, where new test cases are created and obsolete test cases are deleted, the RETECS method learns to prioritize test cases, which are most likely to fail, higher under the guidance of a reward function and by observing previous CI cycles.

Keywords: Software Testing; Machine Learning; Continuous Integration

1 Extended Abstract

Testing in CI requires tight control over the selection and prioritization of the most promising test cases. By most promising, we mean test cases that are prone to detect failures early in the process. In absence of traceability links between code and test cases and based on the hypothesis that test cases having failed in the past are more likely to fail in the future, *history-based test case prioritization* schedules these test cases first in new CI cycles. Testing in CI also means to control the time required to execute a complete cycle. As test case durations vary, not all tests can be executed and *test case selection* is required.

We argue that these two aspects of CI testing can hardly be solved by using only non-adaptive methods. First, the time allocated to test case selection and prioritization in CI is limited. So, time-effective methods shall be privileged over costly and complex prioritization algorithms. Second, history-based prioritization is not well adapted to changes in the execution environment. More precisely, it is frequent to see some test cases being removed from one cycle to another because they test an obsolete feature. At the same time, new test cases are introduced to test new or changed features. Additionally, some test cases are more

[1] Simula Research Laboratory, Lysaker, Norway, firstname@simula.no
[2] University of Stavanger and ABB Robotics, Norway, morten.mossige@uis.no

crucial in certain periods of time, because they test features on which customers focus the most, and then they loose their prevalence because the testing focus has changed. In brief, non-adaptive methods may not be able to spot changes in the importance of some test cases over others because they apply systematic prioritization algorithms.

This paper introduces RETECS, a lightweight test case selection and prioritization approach in CI based on reinforcement learning (RL). RL is well-tuned to design an adaptive method capable to learn from its experience of the execution environment. Our method has the goal to select and prioritize test cases which have been successfully used to detect faults in previous CI cycles, and to order them to execute the most promising ones first. Our method is able to adapt to situations where test cases are added to or deleted from a general repository, while progressively improving its efficiency. It can also adapt to situations where the testing priorities change because of different focus or execution platforms, indicated by changing failure indications. The time required for prioritization is negligible, which is an important aspect in CI. RETECS does not mine code repositories or change-history to compute a new test case schedule. At each cycle, after the tests have been executed, the knowledge to make decisions is updated from feedback provided by a reward function, the only component initially embedding domain knowledge.

The contributions of the paper are threefold: 1) The paper shows that history-based test case prioritization and selection can be approached as a reinforcement learning problem. By modeling the problem with notions such as states, actions, agents, policy, and reward functions, we demonstrate, that RL is suitable to automatically prioritize and select test cases; 2) Implementing an online RL method, without any previous training phase, into a Continuous Integration process is shown to be effective to learn how to prioritize test cases. Comparing two distinct representations (i.e., tableau and neural networks) and three distinct reward functions, our experimental results show that, without any prior knowledge and without any model of the environment, the RL approach is able to learn how to prioritize test cases better than other approaches. Remarkably, the number of cycles required to improve on other methods corresponds to less than 2-months of data, if there is only one CI cycle per day; 3) Experimental results have been computed on industrial data gathered over one year of Continuous Integration. Results show that the method is deployable in industrial settings.

References

[Mo17] Mossige, M.; Gotlieb, A.; Spieker, H.; Meling, H.; Carlsson, M.: Time-Aware Test Case Execution Scheduling for Cyber-Physical Systems. In: Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming. Vol. 10416. LNCS, Springer, pp. 387–404, 2017.

[Sp17] Spieker, H.; Gotlieb, A.; Marijan, D.; Mossige, M.: Reinforcement Learning for Automatic Test Case Prioritization and Selection in Continuous Integration. In: Proceedings of 26th International Symposium on Software Testing and Analysis (ISSTA'17). ACM, pp. 12–22, 2017.