Poster: CUTER: ClUstering-based TEst suite Reduction

Carmen Coviello
University of Basilicata, Italy
carmen.coviello@unibas.it

Simone Romano
University of Basilicata, Italy
simone.romano@unibas.it

Giuseppe Scanniello
University of Basilicata, Italy
giuseppe.scanniello@unibas.it

ABSTRACT

Test Suite Reduction (TSR) approaches speed up regression testing by removing redundant test cases. TSR approaches can be classified as adequate or inadequate. Adequate approaches reduce test suites so that they completely preserve the test requirements (*e.g.,* statement coverage) of the original test suites. Inadequate approaches produce reduced test suites that only partially preserve the test requirements. We propose a tool prototype for inadequate TSR and named it CUTER (ClUstering-based TEst suite Reduction). CUTER implements a clustering-based approach and a number of instances of its underlying process. We implemented CUTER as an Eclipse plug-in and applied it on 19 versions of four Java programs.

CCS CONCEPTS

• **Software and its engineering → Software testing and debugging**;

KEYWORDS

Regression Testing, Test Suite Reduction, Clustering

ACM Reference Format:
Carmen Coviello, Simone Romano, and Giuseppe Scanniello. 2018. Poster: CUTER: ClUstering-based TEst suite Reduction. In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3194983

1 INTRODUCTION

Regression testing is conducted after changes are made to a given System Under Test (SUT) to ensure that these changes do not alter its expected behavior. Test Suite Reduction (TSR) approaches speed up regression testing by removing redundant test cases. They can be classified as adequate or inadequate [3]. Adequate approaches reduce test suites so that they completely satisfy the test requirements (*e.g.,* statement coverage) of the original suite, while inadequate (or non-adequate) approaches only partially preserves the test requirements of the original test suites. Inadequate approaches might be more appealing than adequate approaches if they lead to a higher reduction in test suite size at the expense of a small loss in fault-detection capability [3].

We present CUTER (ClUstering-based TEst suite Reduction) a software prototype intended as an Eclipse plug-in. It implements a Clustering-Based (CB) approach for TSR and a number of instances of the process underlying this approach (simply CB instance/s, from here onwards). CUTER groups together test cases that are similar (and then considered redundant) into a cluster. Test cases are similar if they cover nearly the same statements (*i.e.,* the considered test requirement). To estimate the similarity of test cases, CUTER considers a number of measures (*e.g.,* Euclidean distance), each of which corresponds to a CB instance. CUTER selects, for each cluster, the test case covering the highest number of statements in the cluster. The selected test cases constitute the reduced test suite.

2 BACKGROUND

Clustering algorithms group entities into clusters so that entities within a cluster are similar as much as possible one another [2]. Hierarchical Agglomerative Clustering (HAC) is a kind of clustering algorithm that treats each entity as a singleton cluster at the outset and then successively merges pairs of clusters until all the clusters are merged into a single cluster. At each step the most similar clusters are merged according to a predetermined link criterion. The most used criterion is the average-link one; it evaluates the similarity of two clusters on the basis of all the similarities between the entities in these clusters. Independently from the link criterion, the arrangement of the clusters can be visualized as a tree structure named dendrogram. To identify disjoint clusters the dendrogram needs to be cut at some level. The higher the cut level is, the greater the number of clusters is.

To compare the entities to be clustered, several dissimilarity/distance measures (or simply dissimilarity measures, from here onwards) can be applied. We focus here on the measures implemented in CUTER, which are the most used in HAC applications: Euclidean distance, cosine (dis)similarity, Jaccard-based dissimilarity, Hamming distance, Levenshtein distance, and string kernels-based dissimilarity.

3 CLUSTERING-BASED APPROACH

The process implemented in CUTER is a pipeline. A short description of the phases compounding this process follows:

(1) **ComputingDissimilarity.** On the basis of the statements that each test case covers, this phase computes the dissimilarity between all the pairs of test cases using the measures listed in Section 2.
(2) **ClusteringTestCases.** A clustering algorithm uses the dissimilarity between all the pairs of test cases to group together similar test cases. CUTER uses the HAC algorithm with the average-link criterion. Thus, a cut level for the dendrogram has to be chosen. This choice affects clustering results and then the fault-detection capability of reduced test suites, *e.g.,* if the tester is interested in low fault-detection losses, the cut level value should be high.
(3) **ReducingTestSuite.** For each cluster, the most representative test case is chosen. CUTER selects the test case in a cluster that covers the highest number of statements. This test case is the most representative one. The higher the number of covered statements,

Table 1: Usage steps.

STEP	OUTPUT	DESCRIPTION
(1) Selecting SUT	Selected SUT	The tester selects the SUT (*i.e.,* a Java project with JUnit test cases already imported into the Eclipse workspace) on which he/she wants to produce a reduced test suite.
(2) Clustering Test cases	Dendrogram	The tester chooses if he/she wants to either compute or load the statement coverage for the selected SUT. The tester also chooses the dissimilarity measure to be used. CUTER first calculates the dissimilarity for each pair of test cases and then uses the dissimilarity of test cases to group them into clusters by means of the HAC algorithm. The obtained dendrogram is visualized (see Figure 1). CUTER together implements the first two phases of the process described in Section 3.
(3) Reducing Test Suite	Reduced Test Suite	The tester chooses the cut level of the dendrogram (it is expressed in percentage value with respect to the dendrogram height). CUTER cuts the dendrogram and produces the reduced test suite as described in Section 3 (see the third phase). The reduced test suite is visualized as well as the achieved reductions in test suite size and in code (statement) coverage (see Figure 2).
(4) Saving Reduced Test Suite	Saved JUnit Test Suite	The tester chooses the Java package of the selected SUT where the reduced test suite has to be saved. CUTER generates a JUnit test suite (for the reduced test suite) and saves it.

the greater the likelihood that a test case detects a fault is. If more than one test case cover the same number of statements in a cluster, the most representative test case is chosen randomly among these test cases. The set of the most representative test cases constitutes the reduced test suite.

4 CUTER

CUTER consists of five main software components:

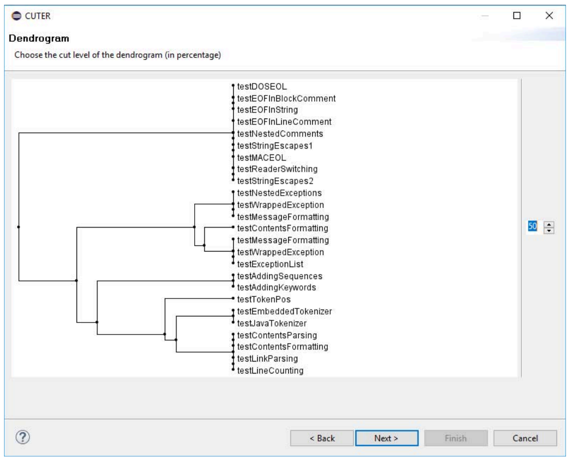
• **Graphical User Interface (GUI).** It enables the interaction of the tester with CUTER. The GUI allows selecting the SUT, computing/loading the statement coverage, selecting the dissimilarity measure, visualizing the dendrogram and choosing its cut level, visualizing and saving the reduced test suite.
• **Computing Statement Coverage**. CUTER exploits JaCoCo[1] and JUnit[2] to compute the statements that each test case covers.
• **Computing Dissimilarity**. Given a dissimilarity measure, this component encodes the covered statements of test cases. Then, it computes the dissimilarity between all the pairs of test cases.
• **Clustering Test Cases**. It implements the HAC algorithm with the average-link criterion. On the basis of the cut level chosen in the GUI, the dendrogram is cut and clusters are obtained.
• **Reducing Test Suite**. It identifies the most representative test case for each cluster. Representative test cases constitute the reduced test suite, which is then implemented as a JUnit test suite.

In Table 1, we report the steps of a use case of CUTER. The output for each step is also shown with a short description. We also provide (when needed) a link between each step and the phases of the process underlying CUTER. A screencast of CUTER in action is available at www2.unibas.it/sromano/CUTER.html.

5 FINAL REMARKS

We proposed an Eclipse plug-in, CUTER, which implements a clustering-based approach for inadequate TSR. It groups together

[1]jacoco.org
[2]junit.org

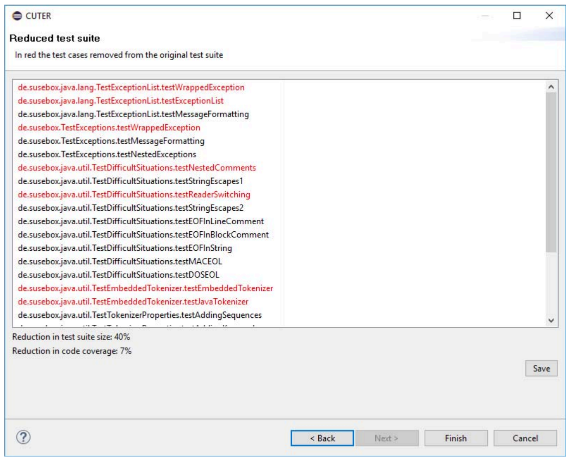


Dendrogram

Choose the cut level of the dendrogram (in percentage)

50

Figure 1: Visualization of dendrogram.



Reduced test suite

In red the test cases removed from the original test suite

```
de.susebox.java.lang.TestExceptionList.testWrappedException
de.susebox.java.lang.TestExceptionList.testExceptionList
de.susebox.java.lang.TestExceptionList.testMessageFormatting
de.susebox.TestExceptions.testWrappedException
de.susebox.TestExceptions.testMessageFormatting
de.susebox.TestExceptions.testNestedExceptions
de.susebox.java.util.TestDifficultSituations.testNestedComments
de.susebox.java.util.TestDifficultSituations.testStringEscapes1
de.susebox.java.util.TestDifficultSituations.testReaderSwitching
de.susebox.java.util.TestDifficultSituations.testStringEscapes2
de.susebox.java.util.TestDifficultSituations.testEOFInLineComment
de.susebox.java.util.TestDifficultSituations.testEOFInBlockComment
de.susebox.java.util.TestDifficultSituations.testEOFInString
de.susebox.java.util.TestDifficultSituations.testMACEOL
de.susebox.java.util.TestDifficultSituations.testDOSEOL
de.susebox.java.util.TestEmbeddedTokenizer.testEmbeddedTokenizer
de.susebox.java.util.TestEmbeddedTokenizer.testJavaTokenizer
de.susebox.java.util.TestTokenizerProperties.testAddingSequences
```

Reduction in test suite size: 40%
Reduction in code coverage: 7%

Figure 2: Reduced test suite (in red test cases that are not included in the reduced test suite).

test cases that are similar. Test cases are similar if they cover nearly the same statements. To estimate such a similarity, CUTER implements a number of measures. A hierarchical agglomerative clustering algorithm is applied to group similar test cases into the same cluster. A reduced test suite will contain a test case for each cluster, namely the test case that covers the largest number of statements. We used CUTER in an empirical study, where we compared the clustering-based approach that CUTER implements with inadequate TSR approaches. The results of this study can be found in Coviello *et al.* [1].

REFERENCES

[1] Carmen Coviello, Simone Romano, Giuseppe Scanniello, Alessandro Marchetto, Giuliano Antoniol, and Anna Corazza. 2018. Clustering Support for Inadequate Test Suite Reduction. In *Proc. of International Conference on Software Analysis, Evolution and Reengineering*. IEEE, 95–105.
[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval.* Cambridge University Press, Cambridge.
[3] August Shi, Alex Gyori, Milos Gligoric, Andrey Zaytsev, and Darko Marinov. 2014. Balancing Trade-offs in Test-suite Reduction. In *Proc. of International Symposium on Foundations of Software Engineering*. ACM, 246–256.