Clustering Support for Inadequate Test Suite Reduction

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

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

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

Alessandro Marchetto
Independent researcher
Email: alex.marchetto@gmail.com

Giuliano Antoniol
Polytechnique Montreal
Montreal, Canada
Email: antoniol@ieee.org

Anna Corazza
University of Naples "Federico II"
Naples, Italy
Email: anna.corazza@unina.it

***Abstract*—Regression testing is an important activity that can be expensive (*e.g.,* for large test suites). Test suite reduction approaches speed up regression testing by removing redundant test cases. These approaches can be classified as adequate or inadequate. Adequate approaches reduce test suites so that they completely preserve the test requirements (*e.g.,* code coverage) of the original test suites. Inadequate approaches produce reduced test suites that only partially preserve the test requirements. An inadequate approach is appealing when it leads to a greater reduction in test suite size at the expense of a small loss in fault-detection capability. We investigate a clustering-based approach for inadequate test suite reduction and compare it with well-known adequate approaches. Our investigation is founded on a public dataset and allows an exploration of trade-offs in test suite reduction. Results help a more informed decision, using guidelines defined in this research, to balance size, coverage, and fault-detection loss of reduced test suites when using clustering.**

I. Introduction

In software maintenance and evolution, 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. Regression testing approaches can be classified into three main classes: *(i)* Test Suite Reduction (TSR); *(ii)* regression test case selection; and *(iii)* test case prioritization [1]. Both TSR and regression test case selection approaches reduce the size of a test suite to speed up regression testing, while preserving its capability to detect faults. TSR approaches delete obsolete or redundant test cases from a test suite [2], while regression test case selection approaches select a subset of test cases to execute only those that exercise changed parts or parts affected by changes of a SUT [3]. TSR is sometimes called test suite minimization when the elimination of test cases is not permanent. However, these two concepts are used in an interchangeable way [1]. Finally, test case prioritization approaches sort test cases according to some criteria [4]. Common to these three classes of approaches is the assumption of the availability of some a-priori knowledge about the SUT and its test suite (*e.g.,* statement coverage).

TSR approaches can be classified as adequate or inadequate [5]. Adequate approaches reduce test suites so that they completely satisfy the test requirements of the original suite. For instance, let statement coverage be the kind of test requirement considered, an adequate TSR approach produces a reduced test suite that covers the same statements as the original test suite. Satisfying all the test requirements does not mean preserving fault-detection capability. A TSR approach is inadequate (or non-adequate) when reduced test suites only partially preserve the test requirements of the original test suites. Inadequate approaches might be more appealing than those adequate if they lead to a greater reduction in test suite size at the expense of a small loss in fault-detection capability [5].

In this paper, we present and investigate a Clustering-Based (CB) approach for TSR and several instances of the process underlying this approach (simply CB instance/s, from here onwards). The CB approach 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. We investigate a number of measures (each of them corresponds to a CB instance) to estimate such a similarity. A reduced test suite will contain a test case for each identified cluster. Given a cluster, the test case that covers the largest number of statements is chosen. It is easy to follow that the CB approach is inadequate because the reduced test suite could not preserve the statement coverage (*i.e.,* the kind of test requirement considered) of the original test suite. We investigate the CB instances with respect to the trade-offs between reductions in test suite size and losses in fault-detection capability. Similarly to Shi *et al.* [5], we also compare our solutions for TSR with well-known adequate approaches. Our investigation is founded on SIR (Software-artifact Infrastructure Repository) [6], a public dataset widely used in the regression testing field (*e.g.,* [7]). Summarizing, we make the following contributions:

Inadequate CB Approach. We investigate the use of clustering to define an inadequate TSR approach, whose underlying process has been instantiated six times. We also define guidelines for these instances to get trade-offs between reductions in test suite size and losses in fault-detection capability.

Inadequate vs Adequate TSR. We compare test suite reductions of adequate approaches with those obtained by applying the CB instances.

Paper Structure. In Section II, we highlight related work, while background in Section III. We describe the CB approach in Section IV. The design of our investigation and the experimental results are provided in Section V and Section VI, respectively. Final remarks conclude the paper in Section VII.

II. Related Work

Rothermel *et al.* [2] formally defined TSR as follows:

Given: A test suite, T, a set of test requirements $r_1, ..., r_n$, that must be satisfied to provide the desired adequate testing of the SUT, and subsets of T, $T_1, ..., T_n$, one associated with each of r_is such that any one of the test cases t_j belonging to T_i can be used to achieve requirement r_i.

Problem: Find a representative set, T', of test cases from T that satisfies all r_is.

Many TSR approaches have been proposed in the literature (*e.g.,* [8]–[14]). Traditional approaches are based on the aforementioned definition and aim at reducing the size of test suites satisfying all the test requirements. Such a kind of approaches is named *adequate* [5]. Researchers have proposed TSR approaches that relax the constraint of satisfying all the test requirements and have named them *inadequate* [5]. In the following subsections, we first summarize the research on adequate and inadequate TSR approaches, then we focus on the use of clustering in the regression testing field.

A. Adequate Approaches

A number of adequate TSR approaches have been proposed [1] in the literature and most of them use heuristic-based criteria to identify and discard redundant test cases. If a test case satisfies a subset of the test requirements of another test case, it could be considered as redundant. Traditional approaches tend to focus on one kind of test requirement (*e.g.,* code coverage) to reduce test suites. For example, Harrold *et al.* [8] proposed a heuristic-based approach to identify a representative set of the original test suite which satisfies all the test requirements. This approach was named HGS (Harrold Gupta Soffa). HGS first analyzes sets of test cases of cardinality one and then chooses the set that satisfies more test requirements. Then, sets of test cases of cardinality two are analyzed and the set that satisfies more requirements is chosen. This process is iteratively performed until all the test requirements are satisfied. Li *et al.* [9] used an approach based on a greedy algorithm (GRD) to prioritize test cases. It was also applied to reduce test suites (*e.g.,* [15]). GRD selects each time a test case, among those available, that maximizes the satisfied test requirements. The selection process concludes when the selected test cases satisfy all the test requirements. Li *et al.* [9] also studied the use of the 2-Optimal algorithm (2OPT) that represents an instance of the K-Optimal algorithm with $K = 2$. The K-Optimal algorithm selects the first K test cases that together satisfy the largest number of test requirements. Tallam and Gupta [10] proposed a variant of GRD named Delayed Greedy (DGR). It is based on the following two steps: *(i)* if a set of test requirements satisfied by a test case t_i is a super-set of a set of requirements satisfied by another test case t_j, then t_j is removed from the test suite; and *(ii)* if a set of test cases satisfying a requirement r_i is a subset of a set of test cases satisfying r_j, then r_i is removed from the test suite.

Most of the TSR approaches are *single-objective* because they consider only one kind of test requirement (*e.g.,* [1], [16]). However, researchers have proposed approaches that consider more than one kind of test requirement. Such a kind of approaches are *multi-objective*. Smith *et al.* [15] proposed a two-objective version of HGS, GRD, 2OPT, and DGR that reduce test suites by using the ratio of code coverage to test case execution cost. Their results suggested that the two-objective versions outperform the single-objective ones.

B. Inadequate Approaches

Shi *et al.* [5] presented a study on some traditional adequate approaches (*e.g.,* HGS and GRD) and their inadequate versions to verify the benefits that can be achieved when relaxing the constraint of covering all the test requirements. The studied inadequate approaches, instead of producing reduced test suites that satisfy 100% of the test requirements, produce reductions satisfying a fixed percentage of the test requirements. The authors observed that for adequate approaches, the size of the reduced test suites is (median of) 62.9% of the size of original test suites. Moreover, by dropping down to 95% of the test requirements (*i.e.,* losing 5% of the covered statements), the reduction in size of the reduced test suites can increase 17.14%.

C. Clustering and Regression Testing

Clustering has been used to group together similar test cases into the same cluster on the basis of different kinds of test requirements (*e.g.,* code coverage and execution cost) [17]–[22]. Similar test cases within a cluster are considered redundant. To reduce test suites, it is needed to select one test case (or more) from each cluster. For example, Parsa *et al.* [17] and Khalilian and Parsa [18] proposed an approach to group test cases based on the similarity of their execution profiles. These authors applied a heuristic method for sampling test cases from clusters so long as the coverage of the reduced test suite is equal to the coverage of the original test suite. Prasad *et al.* [20] designed a CB approach to reduce test suites. This approach uses a method coverage matrix to group test cases with a hierarchical clustering algorithm using Hamming distance. Each cluster is then analyzed to identify subsets of redundant test cases that are candidates for the elimination. These subsets of redundant test cases are selected by analyzing the similarity among: *(i)* functional flows, *(ii)* line coverage, and *(iii)* branch coverage of test cases.

Although TSR and test case prioritization are two different regression testing problems, they can be considered related to one another [1]. For instance, one way of selecting (or minimizing) a set of n test cases from a test suite would be to prioritize the whole set and choose the first n test cases in priority order [1]. Approaches that work in such a way should not guarantee adequacy of reductions because some test requirements could not be satisfied. For example, Carlson *et al.* [22] proposed a prioritization approach based on hierarchical agglomerative clustering with the average-link criterion. The similarity between test cases is based on four kinds of test requirements considered separately: code coverage, code complexity, fault history, and combination of code complexity and fault history. The authors used the Euclidean distance to compute the similarity among test cases for each kind of test requirements. The use of code complexity seemed to produce better prioritizations of test cases. Differently, Arafeen and Do [21] used three types of information to prioritize test cases: functional requirements, code complexity, and importance of functional requirement.

Our proposal can be classified as an approach for inadequate TSR. Differently from the approaches shown before, it is based on a process that can be easily customized to reduce test suites. Our approach enables the identification of clusters of test cases that are similar as regards a kind of test requirement (*i.e.,* test cases in a cluster can be considered redundant with respect to that requirement) using a suitable distance/dissimilarity measure (*e.g.,* Euclidean) and a clustering algorithm (*e.g.,* hierarchical agglomerative). In addition, we compare our approach with adequate TSR approaches (*i.e.,* HGS, GRD, 2OPT, and DGR; and for each of them we consider two versions: single- and two- objective). The results of our investigation help a more informed decision, using guidelines defined in this research, to balance size, coverage, and fault-detection loss of reduced test suites when using clustering.

III. Background

In this section, we provide background information useful to better understand our research.

A. Clustering

Clustering algorithms group entities into clusters (*i.e.,* groups) so that entities within a cluster are similar as much as possible one another. On the other hand, an entity within a given cluster is dissimilar as much as possible to the entities within other clusters [23]. When clustering, a distance/dissimilarity measure has to be chosen. The extent to which two entities are similar (or dissimilar) depends on the chosen measure and its use influences clustering results [23].

Different kinds of clustering algorithms have been proposed in the literature. 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 (or agglomerates) pairs of clusters until all clusters are merged into a single cluster. At each step the most similar clusters are merged. There are different criteria to compute the similarity of two clusters. The average-link criterion evaluates the similarity of two clusters on the basis of all the similarities between the entities in these clusters. That is, in this criterion a pair of clusters with the highest cohesion (*e.g.,* it could be computed as the average similarity of all the pairs of entities in the two clusters) is merged at each iteration. This avoids the pitfalls of the single-link (the similarity of two clusters is the similarity of their most similar entities) and complete-link (the similarity of two clusters is the similarity of their most dissimilar entities) criteria that equate cluster similarity with the similarity of a single pair of entities [23].

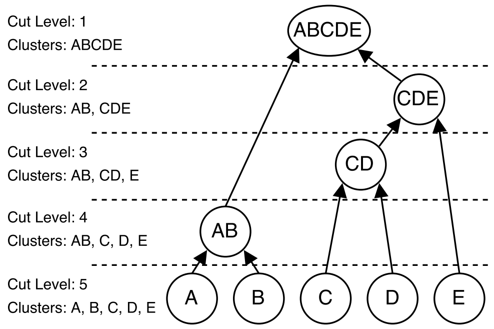


Fig. 1: A sample dendrogram.

The arrangement of the clusters produced by HAC can be visualized as a tree structure named dendrogram (see Figure 1). A HAC algorithm does not require a pre-specified number of clusters [23]. However, in some applications we want a partition of disjoint clusters. In these cases, the dendrogram needs to be cut at some level. This is called cut level of the dendrogram and influences clustering results. In Figure 1, five different cut levels are shown as well as the obtained clusters. For example, choosing three as cut level, the algorithm identifies two clusters containing two entities each and one singleton cluster: AB, CD, and E. The higher the cut levels, the greater the number of clusters is. HAC is more frequently used than top-down (divisive) hierarchical clustering [23].

Hierarchical clustering is preferred in a number of applications because its output is deterministic and the results are arranged in a dendrogram that is more informative than the unstructured set of clusters returned by flat clustering (*e.g.,* k-means).

B. Distance/Dissimilarity Measures

Euclidean Distance. Let $\vec{x} = (x_1, x_2, ..., x_n)$ and $\vec{y} = (y_1, y_2, ..., y_n)$ be two vectors, the Euclidean distance between them is defined as:

$$d(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n} (y_{\text{i}} - x_{\text{i}})^2} \tag{1}$$

Cosine (Dis)similarity. Let $\vec{x}$ and $\vec{y}$ be two vectors, the cosine (dis)similarity between them is defined as:

$$d(\vec{x}, \vec{y}) = 1 - \frac{\vec{x} \cdot \vec{y}}{\|\vec{x}\| \|\vec{y}\|} \tag{2}$$

Jaccard-Based Dissimilarity. Let $\mathbf{A}$ and $\mathbf{B}$ be two sets, the Jaccard-based dissimilarity between them is equal to one minus the Jaccard coefficient:

$$d(\mathbf{A}, \mathbf{B}) = 1 - \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|} \tag{3}$$

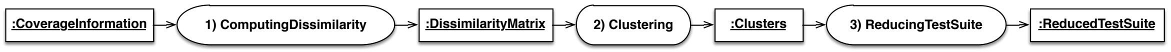


Fig. 2: Process underlying the CB approach for TSR.

Hamming Distance. Let $\vec{x}$ and $\vec{y}$ be two vectors of the same length, the Hamming distance between them is the number of elements in which they differ. For example, given $\vec{x} = (\mathbf{1}, 1, 0, \mathbf{1})$ and $\vec{y} = (\mathbf{0}, 1, 0, \mathbf{0})$ their Hamming distance is two.

Levenshtein Distance. Let s_1 and s_2 be two sequences of characters (*i.e.,* two strings), the Levenshtein distance between them is the minimum number of operations required to transform s_1 into s_2. The operations that can be performed on a sequence are: *(i)* add a new character; *(ii)* delete a character; and *(iii)* substitute a character with another. Different weights can be assigned to each of these operations. We used the following weights: 1 for adding and deleting a character and 2 for the replacement of a character.

String Kernels-Based Dissimilarity. The kernel is a function that computes the inner product between two vectors in their space [24]. In the machine learning field, a string kernel is a particular type of kernel, which operates on strings. Intuitively, they are functions that measure the similarity between strings. The main idea behind these functions is to calculate a dot product between two strings by counting the occurrences of common substrings in the two string. Substrings do not need to be contiguous and can be weighted according to the matching length. String kernels have been designed for classification of proteins and DNA sequences (coded as strings), where mismatches are allowed in matching the substrings between the two strings/sequences [25]. String kernels have not been investigate in software maintenance and evolution.

Giving two strings a and b, the higher the level of similarity between them, the higher the value obtained by computing the string kernel function (*i.e.,* $k(a, b)$). The standard formula is:

$$k(a, b) = \sum_{s \in A^+} num_s(a)\ num_s(b)\ \lambda_s \tag{4}$$

where A^+ represents the set of non-empty substrings, λ_s is a weight or decay factor (it has to be chosen and can be either the same for every substring s or different for each s), $num_s(x)$ is the number of occurrences of the substring s in x (where x is either a or b). In our work, we consider the following instantiation of the kernel function: *boundedrange kernel*, where $\lambda_s = 0$ for all $|s| > n$ and n is fixed. In other words, the boundrange kernel considers only matching substrings of length less than or equal to n.

The string kernels-based dissimilarity between two strings a and b is computed as follows:

$$d(a, b) = 1 - k^*(a, b) \tag{5}$$

where $k^*(a, b)$ is the normalized string kernel function that returns values in between 0 and 1. In our work, we consider the normalized version of the boundrange kernel.

IV. Clustering-based Approach

In this section, we describe the CB approach for inadequate TSR. The process underlying this approach (*i.e.,* a pipeline) is depicted in Figure 2 by means of a UML activity diagram with object flow. Rounded rectangles represent the phases of the process, while rectangles are the objects produced/consumed in these phases. This approach is general and enables the identification of groups of test cases that are similar as regards a kind of test requirement (*e.g.,* code coverage) using a suitable measure (*e.g.,* Euclidean) to compute the similarity between test cases and a clustering algorithm (*e.g.,* flat and hierarchical). In this paper, we present six instances of this process to identify groups of test cases that are similar as regards to statement coverage. The instances are based on the measures described in Section III-B and on a HAC algorithm. A description of the phases of the CB approach follows:

1) **ComputingDissimilarity.** Depending on the kind of test requirement considered, this phase computes the distance/dissimilarity (or simply dissimilarity, from here onwards) between all the pairs of test cases to produce a `DissimilarityMatrix`. This matrix is m x m, where m is the number of the test cases in the test suite, and each entry contains the value of the distance/dissimilarity measure computed on the corresponding pair of test cases. To compute these values, a distance/dissimilarity measure has to be properly chosen and its use needs the definition of a test requirement encoding. The set of statements a given test case covers is encoded as a binary vector in the cases of the Euclidean distance, the cosine (dis)similarity, and the Hamming distance. The dimension of this vector is equal to the number of statements in the SUT. If a statement is not covered then the corresponding element in the vector is equal to zero, one otherwise. To compute the values for the Euclidean distance, the cosine (dis)similarity, and the Hamming distance, we applied the formulations shown in Section III-B. As for the Jaccard-based dissimilarity, we represent the statements that each test case covers as a set (see Section III-B). Finally, for the Levenshtein distance and the string kernels-based dissimilarity (see Section III-B), covered statements are encoded in a string (*i.e.,* a character represents a covered statement).

2) **Clustering.** Test cases are grouped using a clustering algorithm. We opted for the HAC algorithm with the average-link criterion (the motivations behind this decision are those sketched in Section III-A). The used clustering algorithm builds a dendrogram using the `DissimilarityMatrix` produced in the previous phase. Different cut levels on a given dendrogram produce different clustering results. For example, if the tester would obtain small-sized test suite the value of the cut level must be lower (*i.e.,* the number of clusters will be smaller). This could imply a high fault-detection loss. Conversely, if the tester is interested in a low fault-detection loss, the value for the cut level should be higher. Chosen the cut level, this phase produces a set of clusters (*i.e.,* `Clusters`) each containing one or more test cases. The test cases in a cluster are similar as regards the covered statements.

3) **ReducingTestSuite.** For each cluster, the most representative test case has to be chosen. The set of the most representative test cases constitutes the reduced test suite (*i.e.,* `ReducedTestSuite`). To choose the most representative test case different strategies might be used. In our case, the test case in a cluster that covers the largest number of statements is the most representative test case of that cluster. 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. Changing the strategy to select the representative test case of a cluster does not alter the process underlying the CB approach. The exploration of different strategies might represent a possible direction for our research.

V. Empirical Assessment

To conduct our empirical investigation (*i.e.,* a controlled experiment), we followed the guidelines by Wohlin *et al.* [26].

A. *Definition and Context*

The formulated research questions are:

RQ1 - Which is the best instance of the CB approach in terms of trade-off between reductions in test suite size and losses in fault-detection capability?

RQ2 - Is there an instance of the CB approach that outperforms the adequate TSR approaches in terms of trade-off between reductions in test suite size and losses in fault-detection capability?

To perform our experiment, we took into account 19 versions (*i.e.,* experimental objects) of four software systems implemented in Java: eight versions for Ant, three versions for JTopas, five versions for JMeter, and three versions for XMLSecurity. Information on the experimental objects are shown in Table I. The first column of this table reports the name of the systems, while the studied versions are shown in the second column. LOC (Lines of Code) and number of test cases and faults are reported in the last three columns. We selected these experimental objects from SIR,[1] a repository of software-related artifacts whose goal is to support rigorous controlled experiments on regression testing [6]. The selected experimental objects were the same as those used by Zhang *et al.* [7]. Using SIR allows having information on the (hand-seeded) faults of each experimental object. Such information is necessary to estimate the losses in fault-detection capabilities of TSR approaches. SIR has been used in several experiments on regression testing (*e.g.,* [27]–[29]).

B. *Planning*

In our investigation, we studied the following adequate approaches: HGS [8], GRD [9], DGR [10], and 2OPT [9]. For each of these approaches, we considered two versions: single- and two- objective. The single-objective version considers statement coverage as test requirement, while the two-objective version combines statement coverage and test case execution cost (estimated as the time, in milliseconds, to execute a test case). We chose these approaches and their versions because they represent the standard for comparison in the TSR field (*e.g.,* [5], [7], [15], [16]).

[1]sir.unl.edu

TABLE I: Information on the selected experimental objects.

System	Version	LOC	# Test cases	# Faults
Ant	v1 (1.3)	23,796	133	3
	v2 (1.4)	37,478	212	1
	v3 (1.4.1)	37,554	217	2
	v4 (1.5)	64,445	533	8
	v5 (1.5.2)	66,085	539	7
	v6 (1.5.3-1)	66,144	570	1
	v7 (1.6beta)	88,414	842	10
	v8 (1.6beta2)	88,449	845	2
JTopas	v1 (0.4)	4,276	26	5
	v2 (0.5.1)	4,520	28	4
	v3 (0.6)	10,117	56	5
JMeter	v1 (1.8)	33,620	51	4
	v2 (1.8.1)	33,290	63	3
	v3 (1.9.RC1)	37,474	78	8
	v4 (1.9.RC2)	38,613	78	2
	v5 (1.9)	40,989	91	2
XMLSecurity	v1 (1.0.4)	21,601	94	5
	v2 (1.0.5D2)	27,990	94	6
	v3 (1.0.71)	19,731	84	3

As for inadequate TSR approaches, we studied the six instances of the CB approach described in Section IV. Each instance is based on one of the following measures: Euclidean distance (Eucl), cosine (dis)similarity (Cos), Jaccard-based dissimilarity (Jacc), Hamming distance (Hamm), Levenshtein distance (Lev), and string kernels-based dissimilarity (SK). Comparing these instances with a given clustering algorithm (*i.e.,* HAC in our case) would allow identifying the distance/dissimilarity measures more suitable for TSR.

As mentioned in Section III-A, the choice of a cut level of a dendrogram influences the size and composition of the reduced test suite. We express the cut level in percentage value with respect to the height of the dendrogram. A value close to 100% produces more clusters (*i.e.,* we cut the dendrogram at the lower part), while a value close to 0% less clusters. Different criteria might be defined and used to choose cut levels. We consider the following three cut criteria:

1) **Best cut level.** Given a SUT and an instance of the CB approach, the best cut level (for that instance) is the cut level that allows obtaining the highest reduction in test suite size with the lowest reduction in fault-detection capability.

2) **Experimental cut level.** Given the SUTs used in our experiment and an instance of the CB approach, the experimental cut level (for that instance) is the cut level common to all these SUTs that allows obtaining the highest reduction in test suite size with the lowest reduction in fault-detection capability.

3) **RC-based cut level.** Let RC be the Reduction in statement Coverage computed as follows:

$$RC = \frac{C - C_R}{C} \times 100 \tag{6}$$

where C and C_R are the number of statements covered by the original and reduced test suites, respectively. RC assumes values in between 0% and 100%, *e.g.,* a value equal to 5% means that the reduced test suite covers 95% of the statements that the original test suite covers. Given a SUT and an instance of the CB approach, the RC-based cut level is the cut level that allows having a fixed reduction in coverage with respect to the original test suite. On the basis of the results reported by Shi *et al.* [5], we fixed RC to 5% in our investigation. As future work, we will investigate different values for RC.

To identify the cut levels that satisfy the criteria described just before, we cut the dendrogram, built by each CB instance on each experimental object, at any level. Then, we analyzed the results, *i.e.,* reductions in test suite size and losses in fault-detection capability. For space reason, we do not report raw data in the paper, but we made them available on the web.[2]

C. Independent and Dependent Variables

In this experiment, we have one independent variable (or main or manipulated factor) that assumes a value indicating the studied TSR approach/instance. We named this variable **Method**. It assumes the following possible values: CB_X, HGS_Y, GRD_Y, DGR_Y, and $2OPT_Y$. Where X denotes the distance/dissimilarity measure used in our CB approach. For example, CB_{EUCL} indicates the instance of CB with the Euclidean distance. On the other hand, Y denotes either the single-objective (*i.e.,* statement coverage) version or the two-objective (*i.e.,* ratio of statement coverage to test case execution cost) version of the studied adequate approaches: COV and RAT, respectively. For example, HGS_{COV} indicates the single-objective version of the HGS approach.

In prior empirical studies, TSR approaches have been mainly evaluated on the basis of the reduced test suites they produce. In particular, two metrics have been commonly used in these empirical evaluations [1], [2], [13], [30]: Reduction in test suite Size (**RS**) and Reduction in Fault-detection capability (**RF**). Let $|T|$ and $|T_R|$ be the size of the original and reduced test suites, respectively. Let F be the number of faults detected by the original test suite, T, while let F_R be the number of faults detected by the reduced test suite, T_R. RS and RT are defined as follows:

$$RS = \frac{|T| - |T_R|}{|T|} \times 100 \qquad RF = \frac{F - F_R}{F} \times 100 \qquad (7)$$

Both RS and RF assume values in between 0% and 100%. RS reflects the number of test cases within the original test suite that are not present in the reduced one. A value equal to 0% means that the size of the original test suite has not been reduced. Thus, a high value for RS is desirable. As for RF, it is the percentage of faults that the reduced test suite does not detect with respect to the faults the original test suite detects. An RF value equal to 0% means that reduced test suite has preserved the same capability of detecting faults as the original test suite. Thus, a value equal to 0% is desirable. RS and RF are the two analyzed dependent variables.

D. Hypotheses formulation

To test RQ1, we formulated the following null hypotheses:

$NH1_{RS}$ - There is no statistically significant difference in the RS values computed by applying CB_{EUCL}, CB_{COS}, CB_{JACC}, CB_{HAMM}, CB_{LEV}, and CB_{SK}.

$NH1_{RF}$ - There is no statistically significant difference in the RF values computed by applying CB_{EUCL}, CB_{COS}, CB_{JACC}, CB_{HAMM}, CB_{LEV}, and CB_{SK}.

As for RQ2, the formulated null hypotheses are:

$NH2_{RS}$ - There is no statistically significant difference in the RS values computed by applying the instances of the CB approach and HGS_{COV}, GRD_{COV}, DGR_{COV}, $2OPT_{\text{COV}}$, HGS_{RAT}, GRD_{RAT}, DGR_{RAT}, and $2OPT_{\text{RAT}}$.

$NH2_{RF}$ - There is no statistically significant difference in the RF values computed by applying the instances of the CB approach and HGS_{COV}, GRD_{COV}, DGR_{COV}, $2OPT_{\text{COV}}$, HGS_{RAT}, GRD_{RAT}, DGR_{RAT}, and $2OPT_{\text{RAT}}$.

If a null hypothesis is rejected, we can accept the alternative one. For example, if we reject $NH1_{RS}$, we can assume that there is a statistically significant difference in the RS values computed by the six instances of the CB approach.

E. Data Analysis

For each dependent variable and each cut criterion (*e.g.,* experimental cut level), we performed the following steps:

- We computed descriptive statistics and used graphical representations to summarize data (*i.e.,* boxplots).
- We tested each null hypothesis by means of a Kruskal-Wallis test [26]. If the null hypothesis was rejected, we performed a post-hoc analysis. That is, we carried out a pairwise comparison between approaches/instances by applying a two-sided Mann-Whitney test [26]. Both Kruskal-Wallis and Mann-Whitney tests are non-parametric and are well-known for their robustness and sensitiveness [26]. As it is customary, we decided to accept a probability of 5% of committing type-I-error (*i.e.,* $\alpha = 0.05$). When needed, we applied the Bonferroni correction method [31]. For each test, we report exact p-values since this practice makes for greater scientific integrity.

Tests of significance check the presence of statistically significant differences, but they do not provide the magnitude of these differences. Effect size measures are used in statistics to estimate these differences. In our data analysis, we computed the Cliff's δ effect size [32] because it can be used without verifying that data are normally distributed. The magnitude of a difference is considered as: negligible (**N**) if $\delta < 0.147$, small (**S**) if $0.147 \leq |\delta| < 0.33$, medium (**M**) if $0.33 \leq |\delta| < 0.474$, or large (**L**) otherwise [33].

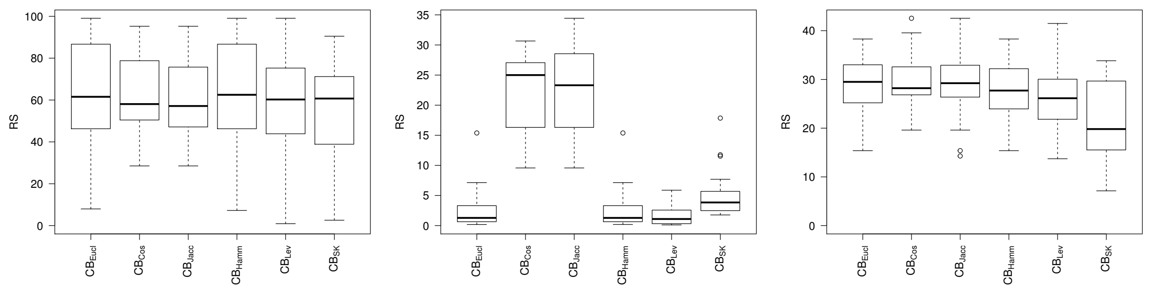
F. Instrumentation

We developed a Java prototype of a supporting software system for the six instances of the CB approach. As for the adequate TSR approaches, we used RAISE, a Java application that implements HGS, GRD, 2OPT, DGR in the single- and two- objective versions. RAISE is available on the web[3] and has been used in previous empirical studies on TSR approaches (*e.g.,* [15], [34]). To gather coverage information, we exploited JaCoCo.[4]

[2]http://www2.unibas.it/sromano/downloads/CUTER_Raw_Data.zip

[3]code.google.com/archive/p/raise/

[4]jacoco.org



(a) Best cut level (b) Experimental cut level (c) RC-based cut level ($RC = 5\%$)

Fig. 3: Boxplots of the RS values of the CB instances (*i.e.,* CB_{EUCL}, CB_{COS}, CB_{JACC}, CB_{HAMM}, CB_{LEV}, and CB_{SK}).

TABLE II: Descriptive statistics (*i.e.,* minimum, maximum, median, mean, and standard deviation) for RS, RF, and RC for each criterion and each instance of the CB approach.

		Best cut level						Experimental cut level						RC-based cut level ($RC = 5\%$)					
		CB_{EUCL}	CB_{COS}	CB_{JACC}	CB_{HAMM}	CB_{LEV}	CB_{SK}	CB_{EUCL}	CB_{COS}	CB_{JACC}	CB_{HAMM}	CB_{LEV}	CB_{SK}	CB_{EUCL}	CB_{COS}	CB_{JACC}	CB_{HAMM}	CB_{LEV}	CB_{SK}
RS	min	7.98	28.52	28.52	7.24	0.93	2.6	0.18	9.57	9.57	0.19	0.12	1.78	15.38	19.61	14.29	15.38	13.73	7.14
	max	99.06	95.24	95.24	99.06	99.06	90.48	15.38	30.66	34.43	15.38	5.88	17.86	38.3	42.55	42.55	38.3	41.49	33.84
	median	61.54	58.06	57.14	62.5	60.26	60.71	1.28	25	23.31	1.28	1.1	3.85	29.5	28.21	29.23	27.72	26.14	19.82
	mean	60.9	63.42	62.44	61.05	57.12	56.2	2.77	27.06	22.03	2.78	1.64	5.32	28.84	29.76	28.93	27.97	26.34	21.21
	SD	26.54	18.63	18.71	26.63	29.41	25.52	3.69	7.04	7.59	3.68	1.71	4.22	5.96	6.18	7.39	6.06	6.63	8.93
RF	min	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	max	0	0	0	0	0	0	0	0	0	0	0	0	14.29	12.5	12.5	14.29	20	14.29
	median	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	mean	0	0	0	0	0	0	0	0	0	0	0	0	0.75	0.66	0.66	0.75	1.81	0.75
	SD	0	0	0	0	0	0	0	0	0	0	0	0	3.28	2.87	2.87	3.28	5.49	3.28
RC	min	0.9	4.32	4.32	0.79	0.12	0.3	0	0.93	0.93	0	0	0.04	2.73	3.54	3.6	2.73	2.91	2.37
	max	73.29	52.71	51.74	73.26	73.29	68.4	2.73	9.31	11.17	2.73	1.34	12.11	6.62	6.15	6.53	6.62	7.14	7.33
	median	17.84	20.61	20.61	17.84	16.87	24.16	0.12	3.76	3.6	0.12	0.1	0.44	5.06	5.08	5.08	5.04	4.95	5.01
	mean	23.3	24.78	23.8	23.52	22.4	27.34	0.38	3.73	4.18	0.38	0.21	1.69	5.14	5.03	5.02	4.99	4.94	5.05
	SD	18.74	15.45	14.07	18.82	18.79	16.96	0.68	2.12	3.07	0.68	0.33	3.07	0.83	0.72	0.83	0.79	0.99	1.29

G. Threats to Validity

To understand strengths and limitations of our investigation, we discuss threats that could affect the validity of the results.
- *Construct validity* threats concern the relationship between theory and observation. In our study, construct validity threats are due to the metrics and dataset. However, the used metrics are widely adopted to quantify the investigated constructs and represent the standard for the assessment of TSR approaches (*e.g.,* [2], [13], [30]). As for the dataset, it has been created by external researchers [6] and has been used in several studies on regression testing (*e.g.,* [27]–[29]).
- *Internal validity* threats concern factors internal to the investigation. How the approaches have been implemented might threaten the validity of the results.
- *Conclusion validity* threats concern the relationship between the dependent and independent variables. The used statistical test might threaten this kind of validity. To deal with this threat, we applied robust and sensitive statistical tests. The chosen adequate approaches might threaten conclusion validity. These approaches can be considered a natural choice for comparing TSR approaches (*e.g.,* [5], [7], [15], [16]).
- *External validity* threats concern the possibility of generalizing results. Although in our investigation we considered software systems previously used in other studies and these systems cover different application domains, we cannot guarantee that our findings can be generalized to the universe of Java systems. Future work is needed to verify to what extent our findings hold for other experimental objects.

VI. Results and Discussion

In this section, we present and discuss the obtained results. We also present implications and future directions.

A. Answering RQ1

Descriptive Statistics and Data Exploration. In Figure 3, we show the boxplots of the RS values of the CB instances obtained by using the best, the experimental, and the RC-based cut level criteria. The boxplots obtained by applying the best cut level criterion suggest that there is a small difference among the RS values. Indeed, the boxplots for CB_{COS} and CB_{JACC} are less skewed than the boxplots of the other instances. As for the experimental cut level, we can observe that CB_{COS} and CB_{JACC} reduce much more than the other instances. Finally, the boxplots for the RC-based cut level criterion are similar one another with only exception of CB_{SK} that seems to reduce less.

In Table II, we report, the values of the descriptive statistics obtained by each CB instance for the dependent variables RS and RF with respect to the criteria: best, experimental, and RC-based cut level. We also report RC values in the last row. We report these values to discuss later on the effect of the trade-off between reductions in test suite size and losses in fault-detection capability on coverage reduction. It is worth noting that, for the RC-based cut level criterion, the actual RC values may be slightly higher/lower than 5% because there could not be a cut level of the dendrogram that allows achieving the exact reduction of 5% in code (statement) coverage. In these cases, we have selected the cut levels that produce RC values closer to 5%.

As for the best cut level, there is not a huge difference among the CB instances with respect to RS (see Table II). CB_{COS} and then CB_{JACC} are slightly better in terms of RS (average values are 63.42% and 62.44%, respectively) than the other CB instances. RF values are always equal to 0% for

TABLE III: Results from the Kruskal-Wallis test for RQ1 (upper part) and RQ2 (lower part). When a p-value is less than α (0.05), it is reported in bold. The p-values for $HN1_{RF}$ and $HN2_{RF}$ in some cases cannot be computed because the RF distributions contain only values equal to 0%.

Hypothesis	Best cut level	Experimental cut level	RC-based cut level ($RC = 5\%$)
$NH1_{RS}$	0.994	**1.58E-16**	**0.0317**
$NH1_{RF}$	-	-	0.9722
$NH2_{RS}$	**1.64E-35**	**4.03E-27**	**7.9E-37**
$NH2_{RF}$	-	-	0.5804

TABLE IV: Results from the pairwise comparisons among CB instances on RS for the experimental cut level criterion. We report in bold p-values less than 0.0083 (*i.e.,* α value normalized by applying the Bonferroni correction).

	CB_{EUCL}	CB_{Cos}	CB_{JACC}	CB_{HAMM}	CB_{LEV}	CB_{SK}
CB_{EUCL}		○ **2.76E-07** L (-0.9778)	○ **2.77E-07** L (-0.9778)	0.9767 -	0.2731 -	○ **0.0030** L (-0.5651)
CB_{Cos}	● **2.77E-07** L (0.9773)		0.895415 -	●**2.77E-07** L (0.9778)	● **1.48E-07** L (1)	● **8.67E-07** L (0.9363)
CB_{JACC}	● **2.76E-07** L (0.9778)	0.8954 -		● **2,76E-07** L (0.9778)	● **1.47E-07** L (1)	● **8.05E-07** L (0.9391)
CB_{HAMM}	0.9767 -	○ **2.77E-07** L (-0.9778)	○ **2.76E-07** L (-0.9778)		0.2425 -	○ **0.003** L (-0.5651)
CB_{LEV}	0.2731 -	○ **1.48E-07** L (-1)	○ **1.47E-07** L (-1)	0.2425 -		○ **0.0002** L (-0.7091)
CB_{SK}	● **0.003** L (0.5651)	○ **8.67E-07** L (-0.9363)	○ **8.05E-07** L (-0.9391)	● **0.003** L (0.5651)	● **0.0002** L (0.7091)	

all the CB instances, *i.e.,* there is no loss in fault-detection capability.

As for the experimental cut level, we obtained similar RS values for CB_{Cos} and CB_{JACC} (see Table II). Indeed, CB_{Cos} is slightly better than CB_{JACC} (*e.g.,* the average RS values are 27.06% and 22.03%, respectively). These two instances produce better RS values as compared with the other instances. RF values are always equal to 0%.

With respect to the RC-based cut level criterion, it seems that there is no CB approach outperforming the others on RS and RF. Indeed, we can observe slightly worse RS values for CB_{SK} as compared with the other instances, while CB_{LEV} is slightly worse on RF.

Hypotheses Testing. In Table III, we report the p-values the Kruskal-Wallis test returned for each tested hypothesis. The reported p-values suggests that we can reject $NH1_{RS}$ when considering the experimental and RC-based cut level criteria (the corresponding p-values are 1.58E-16 and 0.0317). On the other hand, we cannot reject $NH1_{RF}$ because the p-values are either greater than α (RC-based cut level) or the distributions are equal (best and experimental cut levels). These results justify a post-hoc analysis on RS for the experimental and RC-based cut level criteria. The results from the post-hoc analysis are summarized by means on the comparison matrices reported in Table IV and Table V, respectively. An (i, j) entry of these matrices represents the result of the pairwise comparison between the $i-th$ CB approach and the $j-th$ one. In particular, we reported in each entry the p-value returned by the Mann-Whitney test and in case of a statistical significant difference also the Cliff's δ effect size value. Note that the used test is two-sided, thus it indicates if there is or not a statistically significance difference between a pair of CB approaches without indicating which approach significantly outperforms the other. The sign of the effect size values suggests

TABLE V: Results from the pairwise comparisons among CB instances on RS for the RC-based cut level criterion ($RC = 5\%$). We report in bold p-values less than 0.0083 (*i.e.,* α value normalized by applying the Bonferroni correction).

	CB_{EUCL}	CB_{Cos}	CB_{JACC}	CB_{HAMM}	CB_{LEV}	CB_{SK}
CB_{EUCL}		0.8609 -	1 -	0.6507 -	0.1699 -	**0.0079** ● L (0.5069)
CB_{Cos}	0.8609 -		0.8266 -	0.5494 -	0.0932 -	0.0102 -
CB_{JACC}	1 -	0.8266 -		0.5791 -	0.1525 -	0.0173 -
CB_{HAMM}	0.6507 -	0.5494 -	0.5791 -		0.3501 -	0.0195 -
CB_{LEV}	0.1699 -	0.0932 -	0.1525 -	0.3501 -		0.1083 -
CB_{SK}	**0.0079** ○ L (-0.5069)	0.0102 -	0.0173 -	0.0195 -	0.1083 -	

which approach wins the pairwise comparison. To facilitate the reading of the comparison matrices, when there is a statistically significance difference, we indicate the winner of a pairwise comparison with ●, while the loser with ○. For example, the row of CB_{SK} in Table IV indicates that CB_{SK} significantly outperforms CB_{EUCL} (● is reported) and the effect size is large (0.5651), but CB_{SK} is significantly outperformed by CB_{Cos} (○ is reported) and the effect size is large (-0.9363).

When considering the experimental cut level criterion, the results of the pairwise comparisons on RS (see Table IV) suggest that CB_{Cos} and CB_{JACC} are the best CB instances. Indeed, there are statistically significant differences in the RS values between CB_{Cos} and each of the following instances: CB_{EUCL}, CB_{HAMM}, CB_{LEV} and CB_{SK}. These differences are large (values range in between 0.9363 and 1). Similar results are observed for CB_{JACC}. That is, there are statistically significant differences when comparing CB_{JACC} against CB_{EUCL}, CB_{HAMM}, CB_{LEV}, and CB_{SK}. These differences are large (values range in between 0.9391 and 1). The comparison between CB_{Cos} and CB_{JACC} does not have a winner, *i.e.,* there is no statistically significant difference in the RS values they obtained. As for the other instances, the results in Table IV indicate that CB_{HAMM} and CB_{LEV} do not win any pairwise comparison, while CB_{SK} statistically outperforms CB_{EUCL}, CB_{HAMM} and CB_{LEV} (effect size values range from 0.5651 to 0.7091, thus large) but it is outperformed by CB_{Cos} and CB_{JACC} (effect size values are equal to -0.9363 and -0.9391, thus large).

As far as the RC-based cut level criterion is concerned, the results summarized in Table V indicate that there are not statistically significant differences in the RS values except for two cases: CB_{EUCL} significantly outperforms CB_{SK} and the effect size is large (0.5069), and consequently CB_{SK} is outperformed by CB_{EUCL}.

Discussion. On the basis of the observed results, we can answer RQ1 as follows: *the best instances of the CB approach in terms of trade-off between reductions in test suite size and losses in fault-detection capability are CB_{Cos} and CB_{JACC}*. In fact, when considering the experimental cut level criterion, CB_{Cos} and CB_{JACC} reduce the size of the original test suites significantly more than CB_{EUCL}, CB_{HAMM}, CB_{LEV}, and CB_{SK}, without any effect on fault-detection capability. The effect size is large in all these cases. We also observed that the experimental cut level is equal to 75% for both CB_{Cos} and CB_{JACC}. This outcome

TABLE VI: Descriptive statistics for RS and for each version of the studied adequate approaches.

	Single-objective version				Two-objective version			
	HGS_{COV}	GRD_{COV}	DGR_{COV}	$2OPT_{\text{COV}}$	HGS_{RAT}	GRD_{RAT}	DGR_{RAT}	$2OPT_{\text{RAT}}$
min	0	0	0	0	0	0	0	0
max	13.19	13.19	13.19	12.09	13.19	13.79	13.19	9.89
median	0	0	0	0	0	0	0	0
mean	1.59	1.56	1.59	1.49	1.59	1.51	1.59	0.7
SD	3.15	3.16	3.15	2.93	3.15	3.15	3.15	2.34

can be seen as a guideline for the tester, who could choose this cut level being confident of significantly reducing the size of the original test suite without notably losing in fault-detection capability. The tester would be also confident that a reduction in test suite size slightly affects code coverage (see the RC values in Table II). The experimental cut levels for the other CB instances were: 87.5% for both CB_{HAMM} and CB_{EUCL}, 94.3% for CB_{SK}, and 95% for CB_{LEV}.

When considering the RC-based cut level criterion, we observed that the CB_{SK} is significantly worse than CB_{EUCL} with respect to the reduction in size of the original test suite. In all the other pairwise comparisons there is no significant difference in the size and fault-detection capability of the reductions. That is, there is no a clear winner as far as the RC-based cut level criterion is concerned. In this case, the guideline for the tester is not to use CB_{SK} and to choose a cut level that allows having 5% of reduction in code coverage. The application of the RC-based cut level criterion, on average, decreases code coverage more than the experimental cut level.

Finally, for the best cut level criterion the Kruskal-Wallis test did not allow us to reject the defined null hypotheses. In addition, for this criterion we did not find a guideline to choose a proper cut level because we observed remarkable variations in the results achieved on the experimental objects. However, this kind of criterion could not be applied in practice because a tester would not know the reduction in fault-detection capability at the time he/she would apply a given CB instance. We can also point out that the trade-offs (reductions in test suite size and losses in fault-detection capability) for the best cut level criterion are better than the other two criteria. However, the application of the best cut level criterion reduces much more code coverage, so notably affecting test requirements.

B. Answering RQ2

Descriptive Statistics and Data Exploration. In Table VI, for each version of the studied adequate approaches, we report the values of descriptive statistics (*i.e.,* minimum, maximum, median, mean, and standard deviation values) for RS. These values are very similar one another except for $2OPT_{\text{RAT}}$, which reduces slightly worse than the others on average (0.7%). The studied adequate approaches do not lose any fault (*i.e.,* the RF values are always equal to 0%). This is why we do not report descriptive statistics for RF. We can observe that the best adequate approaches produce RS values much worse than those of the best CB instances, *i.e.,* CB_{COS} and CB_{JACC} (this holds for any cut criterion). For example, CB_{COS} with the experimental cut level produces, on average, RS values equal to 27.06%, while the highest average RS values for the adequate approaches are equal to 1.59% (*e.g.,* HGS_{COV}). In both the case, RF values are always equal to 0%.

Hypotheses Testing. The results of the Kruskal-Wallis test are summarized in the lower part of Table III. For any cut criterion, this test allows us to reject $NH2_{RS}$ (p-values are less than α). As for $NH2_{RF}$, in no case we can reject the null hypothesis because the p-values are either greater than α (RC-based cut level) or the distributions are equal (best and experimental cut levels). On the basis of the obtained results, we performed a pairwise comparison between each CB instance and each studied adequate approach on RS. The results (*i.e.,* the p-values returned by the Mann-Whitney test and the effect size values in case of statistically significant differences) are summarized in Table VII. We can observe that, independently from the cut criterion, all the CB instances (except for CB_{LEV} with the experimental cut level) are significantly better (applying the Bonferroni correction) than the adequate approaches on RS and the effect size is always large.

Discussion. On the basis of the observed results, we can positively answer RQ2: *the instances of the CB approach outperform the adequate TSR approaches in terms of trade-off between reductions in test suite size and losses in fault-detection capability.* In other words, the CB approach for inadequate TSR is appealing because it leads to greater reductions in test suite size at the expense of small losses in fault-detection capability. We also observed slight reductions in code coverage when considering the RC-based and experimental cut level criteria, while this is not the case for the best cut level criterion.

C. Implications and Future Extensions

We focus on the researcher and the practitioner perspectives for the discussion of the implications and future extensions for the research presented in this paper.

- We observed that, when using the experimental cut level criterion, CB_{COS} and CB_{JACC} significantly outperform the other CB instances in terms of trade-off between reductions in test suite size and losses in fault-detection capability. We experimentally found that the cut level satisfying this criterion is equal to 75% of dendrogram height for both CB_{COS} and CB_{JACC}. That is, we have found a guideline, for CB_{COS} and CB_{JACC}, so supporting the practitioner (*e.g.,* the software tester) in taking a more informed decision on the cut level to reduce test suites that balances size and fault-detection loss of the reduced test suites slightly affecting the statement coverage. The researcher could be interested to further explore guidelines in order to obtain better trade-offs in test suite reductions. This represents a future direction for our research.
- We found that the CB approach reduces more the test suite size than the studied adequate approaches. For example, the

TABLE VII: Results from the pairwise comparisons between each CB instance and each adequate approach on RS. We report in bold p-values less than 0.0083 (*i.e.,* α value normalized by applying the Bonferroni correction).

		Single-objective version				Two-objective version			
		HGS_{Cov}	GRD_{Cov}	DGR_{Cov}	$2OPT_{\text{Cov}}$	HGS_{RAT}	GRD_{RAT}	DGR_{RAT}	$2OPT_{\text{RAT}}$
Best cut level	CB_{EUCL}	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.1E-07** L (0.9945)	**1.22E-07** L (0.9945)	**5.58E-08** L (0.9945)
	CB_{Cos}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.33E-08** L (1)	**1.04E-07** L (1)	**4.70E-08** L (1)
	CB_{JACC}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.33E-08** L (1)	**1.04E-07** L (1)	**4.70E-08** L (1)
	CB_{HAMM}	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.22E-07** L (0.9945)	**1.1E-07** L (0.9945)	**1.22E-07** L (0.9945)	**5.58E-08** L (0.9945)
	CB_{LEV}	**5.06E-07** L (0.9446)	**5.06E-07** L (0.9446)	**5.06E-07** L (0.9446)	**5.06E-07** L (0.9446)	**5.06E-07** L (0.9446)	**3.93E-07** L (0.9501)	**5.06E-07** L (0.9446)	**7.82E-08** L (0.9838)
	CB_{SK}	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.78E-07** L (0.9778)	**1.98E-07** L (0.9778)	**6.61E-08** L (0.9889)
Experimental cut level	CB_{EUCL}	**2.22E-05** L (0.7978)	**2.22E-05** L (0.7978)	**2.22E-05** L (0.7978)	**1.83E-05** L (0.8061)	**2.22E-05** L (0.7978)	**1.13E-05** L (0.8227)	**2.22E-05** L(0.7978)	**1.03E-06** L (0.8947)
	CB_{Cos}	**2.72E-07** L (1)	**2.72E-07** L (1)	**2.72E-07** L (1)	**2.32E-07** L (1)	**2.72E-07** L (1)	**2.45E-07** L (1)	**2.72E-07** L (1)	**6.61E-08** L (1)
	CB_{JACC}	**1.98E-07** L (1)	**1.98E-07** L (1)	**1.55E-07** L (1)	**1.98E-07** L (1)	**1.98E-07** L (1)	**1.78E-07** L (1)	**5.57E-08** L (1)	**1.98E-07** L (1)
	CB_{HAMM}	**3.51E-05** L (0.7784)	**3.51E-05** L (0.7784)	**3.51E-05** L (0.7784)	**3.51E-05** L (0.7784)	**3.51E-05** L (0.7784)	**2.53E-05** L (0.7784)	**3.51E-05** L (0.7784)	**1.89E-06** L (0.8726)
	CB_{LEV}	0.0547 -	0.0346 -	0.0547 -	0.0346 -	0.0547 -	0.0261 -	0.0547 -	**8.15E-06** L (0.8172)
	CB_{SK}	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**0.0016** L (0.5956)	**6.13E-06** L (0.8283)
RC-Based cut level ($RC = 5\%$)	CB_{EUCL}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.33E-08** L (1)	**1.04E-07** L (1)	**4.70E-08** L (1)
	CB_{Cos}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.32E-08** L (1)	**4.69E-08** L (1)	**1.04E-07** L (1)
	CB_{JACC}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.33E-08** L (1)	**4.70E-08** L (1)	**1.04E-07** L (1)
	CB_{HAMM}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.33E-08** L (1)	**1.04E-07** L (1)	**4.70E-08** L (1)
	CB_{LEV}	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**1.04E-07** L (1)	**9.32E-08** L (1)	**1.04E-07** L (1)	**4.69E-08** L (1)
	CB_{SK}	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.98E-07** L (0.9778)	**1.78E-07** L (0.9778)	**1.98E-07** L (0.9778)	**7.81E-08** L (0.9834)

CB_{Cos} instance (with 75% as cut level) on average reduces test suite size of 27.06%, with a small reduction in code coverage (3.73% on average), while the best adequate approach on average reduces test suite size of 1.59%. In other words, losing 3.73% in code coverage could increase the reduction in test suite size 25.47% without losses in fault-detection capability. Test suite reduction could increase up to 28.17%, when using the RC-based cut level ($RC = 5\%$), with a small effect on fault-detection capability (0.66%, on average). Also Shi *et al.* [5] observed a trend similar to that mentioned just before (see Section II-B for a quick reference). As future extension of our work, we plan to compare our CB approach with existing inadequate approaches for TSR (*e.g.,* [5]).

- To cluster test cases, the use of the cosine and the Jaccard-based dissimilarities seems to be more promising than the use of the Euclidean, Hamming, and Levenshtein distances and the string kernels-based dissimilarity. This result might be of interest for both the researcher and the practitioner who could choose either cosine or Jaccard-based dissimilarity to cluster test cases. In addition the researcher could be interested to investigate different distance/dissimilarity measures. Our study poses the basis for more informed investigations.

- We observed that for the RC-based cut level criterion, there is not an instance of the CB approach that significantly outperforms the others, so suggesting that all the instances (except CB_{SK}) might perform similarly. This outcome is of interest for the practitioner. In particular, he/she can use any of the CB instances (except CB_{SK}) and apply the RC-based cut level criterion being confident of obtaining a good trade-off between reduction in test suite size and loss in fault-detection capability, with a fixed reduction in code coverage.

- The CB approach has been designed to be easy to adapt to different programming languages, test requirements, distance/dissimilarity measures, and clustering algorithms. For example, the researcher could be interested in investigating the CB approach in the context of C++ software systems.

VII. Final Remarks

In this paper, we investigate a clustering-based approach for inadequate TSR. It groups together test cases that are similar. Test cases are similar if they cover nearly the same statements. To estimate such a similarity, we considered a number of measures. A hierarchical agglomerative clustering is applied to group similar test cases into the same cluster. A reduced test suite will contain a test case for each of these identified clusters. The test case that covers the largest number of statements is chosen for each cluster. We investigate our approach and also compare it with well-known adequate approaches. We founded our investigation on a public dataset. Results suggest that our approach is appealing because it leads to a greater reduction in test suite size at the expense of a small loss in fault-detection capability. We also defined guidelines to help a more informed decision about balancing size, coverage, and fault-detection loss of reduced test suites.

References

[1] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: A survey," *Softw. Test. Verif. Reliab.*, vol. 22, no. 2, pp. 67–120, 2012.

[2] G. Rothermel, M. J. Harrold, J. von Ronne, and C. Hong, "Empirical studies of test-suite reduction," *Softw. Test. Verif. Reliab.*, vol. 12, no. 4, pp. 219–249, 2002.

[3] G. Rothermel and M. J. Harrold, "Analyzing regression test selection techniques," *IEEE Trans. Softw. Eng.*, vol. 22, no. 8, pp. 529–551, 1996.

[4] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 159–182, 2002.

[5] A. Shi, A. Gyori, M. Gligoric, A. Zaytsev, and D. Marinov, "Balancing trade-offs in test-suite reduction," in *Proceedings of International Symposium on Foundations of Software Engineering*. ACM, 2014, pp. 246–256.

[6] H. Do, S. G. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact." *Empirical Softw. Engg.*, vol. 10, no. 4, pp. 405–435, 2005.

[7] L. Zhang, D. Marinov, L. Zhang, and S. Khurshid, "An empirical study of junit test-suite reduction," in *Proceedings of International Symposium on Software Reliability Engineering*. IEEE Computer Society, 2011, pp. 170–179.

[8] M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Trans. Softw. Eng. Methodol.*, vol. 2, no. 3, pp. 270–285, 1993.

[9] Z. Li, M. Harman, and R. M. Hierons, "Search algorithms for regression test case prioritization," *IEEE Trans. Softw. Eng.*, vol. 33, no. 4, pp. 225–237, 2007.

[10] S. Tallam and N. Gupta, "A concept analysis inspired greedy algorithm for test suite minimization," in *Proceedings of Program Analysis for Software Tools and Engineering*. ACM, 2005, pp. 35–42.

[11] T. Y. Chen and M. F. Lau, "Dividing strategies for the optimization of a test suite," *Information Processing Letters*, vol. 60, no. 3, pp. 135 – 141, 1996.

[12] J. A. Jones and M. J. Harrold, "Test-suite reduction and prioritization for modified condition/decision coverage," *IEEE Trans. Softw. Eng.*, vol. 29, no. 3, pp. 195–209, 2003.

[13] D. Jeffrey and N. Gupta, "Test suite reduction with selective redundancy," in *Proceedings of International Conference on Software Maintenance*. IEEE Computer Society, 2005, pp. 549–558.

[14] S. K. Mohapatra and S. Prasad, "Minimizing test cases to reduce the cost of regression testing," in *Proceedings of International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE Computer Society, 2014, pp. 505–509.

[15] A. M. Smith and G. M. Kapfhammer, "An empirical study of incorporating cost into test suite reduction and prioritization," in *Proceedings of Symposium on Applied Computing*. ACM, 2009.

[16] D. Jeffrey and N. Gupta, "Improving fault detection capability by selectively retaining test cases during test suite reduction," *IEEE Trans. Softw. Eng.*, vol. 33, no. 2, pp. 108–123, 2007.

[17] S. Parsa, A. Khalilian, and Y. Fazlalizadeh, "A new algorithm to test suite reduction based on cluster analysis," in *Proceedings of International Conference on Computer Science and Information Technology*. IEEE Computer Society, 2009, pp. 189–193.

[18] A. Khalilian and S. Parsa, "Bi-criteria test suite reduction by cluster analysis of execution profiles," in *Proceedings of IFIP TC 2 Central and East European Conference on Advances in Software Engineering Techniques*. Springer-Verlag, 2012, pp. 243–256.

[19] S. Yoo, M. Harman, P. Tonella, and A. Susi, "Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge," in *Proceedings of International Symposium on Software Testing and Analysis*. ACM, 2009, pp. 201–212.

[20] S. Prasad, M. Jain, S. Singh, and C.Patvardhan, "A multi coverage criteria test suite minimization technique," *International Journal of Applied Information Systems*, vol. 1, no. 8, pp. 5–11, 2012.

[21] M. J. Arafeen and H. Do, "Test case prioritization using requirements-based clustering," in *Proceedings of International Conference on Software Testing, Verification and Validation*. IEEE Computer Society, 2013, pp. 312–321.

[22] R. Carlson, H. Do, and A. Denton, "A clustering approach to improving test case prioritization: An industrial case study," in *Proceedings of International Conference on Software Maintenance*. IEEE Computer Society, 2011, pp. 382–391.

[23] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 2008.

[24] A. Karatzoglou and I. Feinerer, "Kernel-based machine learning for fast text mining in r," *Computational Statistics & Data Analysis*, vol. 54, no. 2, pp. 290–297, 2010.

[25] C. Leslie, E. Eskin, J. Weston, and W. Noble, "Mismatch string kernels for svm protein classification," in *Advances in Neural Information Processing Systems 15*. MIT Press, 2003, pp. 1417–1424.

[26] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[27] A. Shi, T. Yung, A. Gyori, and D. Marinov, "Comparing and combining test-suite reduction and regression test selection," in *Proceedings of Joint Meeting on Foundations of Software Engineering*. ACM, 2015, pp. 237–247.

[28] A. Panichella, R. Oliveto, M. Di Penta, and A. De Lucia, "Improving multi-objective test case selection by injecting diversity in genetic algorithms," *IEEE Trans. Software Eng.*, vol. 41, no. 4, pp. 358–383, 2015.

[29] B. Jiang, Z. Zhang, T. H. Tse, and T. Y. Chen, "How well do test case prioritization techniques support statistical fault localization," in *Proceedings of International Computer Software and Applications Conference*. IEEE Press, 2009, pp. 99–106.

[30] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, "Effect of test set minimization on fault detection effectiveness," in *Proceedings of International Conference on Software Engineering*. ACM, 1995, pp. 41–50.

[31] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–64, 1961.

[32] N. Cliff, *Ordinal methods for behavioral data analysis*. Psychology Press, 1996.

[33] J. Romano, J. D. Kromrey, J. Coraggio, and J. Skowronek, "Appropriate statistics for ordinal level data: Should we really be using t-test and Cohen'sd for evaluating group differences on the NSSE and other surveys?" in *annual meeting of the Florida Association of Institutional Research*, 2006, pp. 1–3.

[34] A. M. Smith, J. Geiger, G. M. Kapfhammer, and M. L. Soffa, "Test suite reduction and prioritization with call trees," in *Proceedings of Symposium on Applied Computing*. ACM, 2007, pp. 539–540.