Empirical Investigation of the Effects of Test Suite Properties on Similarity-Based Test Case Selection

Hadi Hemmati[a,b], Andrea Arcuri[a], Lionel Briand[a,b]

[a]Simula Research Laboratory

[b]Department of Informatics, University of Oslo
{hemmati, arcuri, briand} @simula.no

***Abstract*—Our experience with applying model-based testing on industrial systems showed that the generated test suites are often too large and costly to execute given project deadlines and the limited resources for system testing on real platforms. In such industrial contexts, it is often the case that only a small subset of test cases can be run. In previous work, we proposed novel test case selection techniques that minimize the similarities among selected test cases and outperforms other selection alternatives. In this report, our goal is to gain insights into why and under which conditions similarity-based selection techniques, and in particular our approach, can be expected to work. We investigate the properties of test suites with respect to similarities among fault revealing test cases. We thus identify the ideal situation in which a similarity-based selection works best, which is useful for devising more effective similarity functions. We also address the specific situation in which a test suite contains outliers, that is a small group of very different test cases, and show that it decreases the effectiveness of similarity-based selection. We then propose, and successfully evaluate based on two industrial systems, a solution based on rank scaling to alleviate this problem.**

Keywords- Test Case Selection; Similarity Measure; Distance Function; Adaptive Random Testing; Genetic Algorithms; Model Based Testing

1. Introduction

Rewarding diversity in test cases has been shown to lead to higher fault detection in numerical applications, because failing test cases tend to cluster in contiguous regions [1]. In previous work [2-4], we proposed similarity-based test case selection (STCS) techniques for model based testing (MBT), which applied the idea of rewarding diversity on abstract test cases generated from UML state machines.

Our motivation was that running all the system level test cases generated by a standard criterion, e.g., round trip path for UML state machines, was not feasible, due to the high cost of running them on the deployed platform or test network. So, from a practical standpoint, to solve the testing problems of our industrial partners, it was necessary to devise techniques to select smaller test suites. Our approach is, given a small budget of test cases that can be run, to reward diversity (i.e., penalize similarity) in the chosen test cases.

We assessed different similarity measures and diversified test cases using Genetic Algorithms (GAs) and Adaptive Random Testing (ART). The proposed techniques were applied on one industrial case study where the goal was to decrease test execution cost down to an affordable number of test cases while preserving, to the maximum extent, the fault detection rate (FDR) of the original test suite. Results showed that, compared to random and coverage-based selection, much higher FDR can be achieved when using STCS.

These promising results motivated the need to gain a better understanding of STCS, which is essential to develop novel and more effective techniques. Unlike our previous work [2-4], where we were exploring alternative STCS techniques, in this report we analyze their variation in effectiveness, in a controlled manner and using simulation, when varying the relationship between similarity distributions and fault detection among test cases. In other words, the goal is to investigate under which circumstances STCS is more effective. The results shed light on the best and worst conditions for STCS, thus preparing the ground for improved similarity measures and STCS results.

The intuition is that STCS would perform better when test cases which detect distinct faults are dissimilar and test cases that detect a common fault are similar. Such a condition was verified [2] in one industrial case study, where we found that test cases finding a common fault were indeed clustered together in the test case space and these clusters were mostly distinct.

In this report, to investigate the above intuition in a more precise and systematic way, we resort to a large number of experiments based on simulation. Two industrial case studies were used to guide the simulations and thereby obtain more realistic results. On one hand, the results of our empirical study confirm our intuition about what drives the effectiveness of STCS, though they provide insights that are more complex than what was originally expected. On the other hand, such analyses pointed out a particular characteristic of MBT (compared to numerical applications) that can make STCS less effective. The situation appears when there is a small clustered set of test cases that is far away from all the others (referred to as *outliers*), which is not uncommon, for example, in state machine-based testing when a small group of transition paths is mostly disconnected from the rest of the state machine. Our empirical analyses show that, in that case, the FDR of STCS can significantly decrease. We hence propose an approach, based on rank scaling, to manipulate similarity values so as to alleviate this problem.

The rest of the report is organized as follows. The next section provides background information about similarity-based test case selection. Section 3 discusses the problems related to outliers and outlines our solution to alleviate it. Section 4 describes the experiment design and reports the results. Section 5 provides a brief overview of related works covering similarity-based selection techniques. Finally, Section 6 concludes the report and outlines our future work plan.

2. Similarity based test case selection

Unlike coverage-based selection, where the goal is maximizing the coverage of a test model (e.g., transition coverage in a state machine) by the selected test cases to maximize chances of fault detection, STCS techniques maximize diversity among the selected test cases. Diversity is calculated using a (dis)similarity measure between pairs of test cases. A similarity measure is a value that a similarity function assigns to the pair. Inputs of the function are usually an encoded test case as a set or sequence of elements. In the context of MBT, the inputs are abstract test cases defined on the test model rather than concrete test cases. We do not use the execution

information of the test case as, in our context, the goal is to select them before execution. Abstract test cases are naturally generated as a first step by MBT and can hide the unnecessary information for similarity comparisons. For example, in state machine-based testing, an abstract test case representation can be a path in the state machine specifying the software under test (SUT). In general, different faults can be detected by the same test path instantiated with different test data (e.g., event parameter values). Therefore, to calculate the FDR of a technique, it is necessary to run the selected test paths with different input data and analyze its FDR distribution.

2.1. Encoding and similarity functions

The representation (encoding) of test cases has an important effect on a similarity measure. Though in MBT a test path represents an encoded version of a test case, the test path can be described at different levels of details. In [4], we studied three encodings for a test path in UML state machines: state-based, transition-based, and trigger-guard-based. We reported that trigger-guard-based encoding is the most effective one in terms of fault detection, where a test path (*tp*) is represented as:

<tp> ::= <TrGu> | <TrGu> "," <tp>
< TrGu > ::= *trig* | *guard* | *id* | *guard* "+" *trig*

where *trig* is the identification of a trigger, and *guard* is the identification of a guard in the state machine. In this representation, a transition is identified by its trigger, a guard, or both. If there is a transition with no guard and trigger, we use the transition id (*id*) as its identifier.

Given an encoding, one may use different similarity functions to calculate the similarity value. In [3] we studied six set-based and sequence-based similarity functions. We proposed Jaccard Index [5] as the most cost-effective set-based and Needleman-Wunsch (NW) [6] as the best sequence-based similarity function. The Jaccard Index is defined as the size of the intersection divided by the size of the union of the two encoded test cases, whereas the NW algorithm assigns a similarity value based on the global alignment [6] of the two encoded test cases by arranging the sequences of elements in the test cases to identify regions of similarity between the sequences.

From an STCS point of view, the only required constraints on the similarity measures are that they must be positive and symmetric, which is true for all proposed set and sequence based measures. This means that properties such as reflexivity ($Sim(tp_i , tp_j)$=*maximum* iff $tp_i = tp_j$) and triangular inequality ($D(tp_i , tp_j)$+ $D(tp_j , tp_k)$>= $D(tp_i , tp_k)$ where $D(tp_i , tp_j)$ = 1/ $Sim(tp_i , tp_j)$) do not necessarily hold among different pairs of test cases [7]. For example, NW values can be in any range and, except for symmetry, does not feature any other well-known property of distance/similarity measures [8].

Given a set of *n* encoded test cases (s_n) and a similarity function (*Sim*), the test case selection problem is reformulated as minimizing *SimMsr*(s_n):

$$SimMsr(s_n) = \sum_{tp_i , tp_j \in s_n \wedge i>j} Sim(tp_i , tp_j)$$

where $Sim(tp_i , tp_j)$ returns the similarity of two test paths (encoded abstract test cases in MBT) in s_n represented by tp_i and tp_j. The last step in STCS is using a strategy to select a subset of test cases with minimum average pair-wise similarity (*SimMsr*). This test case selection problem is NP hard (traditional set cover) [9] and using an exhaustive search in our cases (and for most realistic cases) is not an option, since the search space size for selecting a subset of size *n* is equal to the number of possible *n*-combinations within a test suite of a given size. As an example from our case studies, the search space size for *n*=28 in a test suite of size 281 (~10% of the test suite) is $\binom{281}{28} \cong 2.9*10\text{^}38$.

In [2] we have examined GA, ART, and a hierarchal clustering algorithm as selection algorithms and found out that GA was the most cost-effective technique among them. In this report, we show our analyses when both using GA as our proposed technique (called STCS_GA) and ART as the most well-known algorithm (called STCS_ART) for diversifying test cases. The algorithms are introduced in the following subsections.

2.2. Adaptive random testing

ART has been proposed as an extension to Random Testing [1]. Its main idea is that diversity among test cases should be rewarded, because failing test cases tend to be clustered in contiguous regions of the input domain. This has been shown to be true in empirical analyses regarding applications whose input data are of numerical type [1]. Therefore, ART is a candidate selection strategy in our context as well. In this report, we use the basic ART algorithm described in [1], but we ensure that no replicated test case is given in output. The pseudo-code for ART is:

(1) Z={}
(2) Add a random test case to Z
(3) Repeat until |Z|= *sampleSize*
(4) Sample K random test cases that are different from Z
(5) For each of these test cases k
(6) k.maxSim = max(Sim(k , z ∈ Z))
(7) Add the k with minimum maxSim to Z

2.3. Genetic Algorithms

In this report, we use a steady state GA with the same settings as it has been used in [2], in which only the offspring that are not worse than their parents are added to the next generations. An individual in our context is a subset of size *n* from the original test suite (denoted s_n). Given a similarity function $Sim(tp_i , tp_j)$, the fitness function *f* to minimize is the sum, for all pairs (tp_i , tp_j) in s_n, of $Sim(tp_i , tp_j)$, denoted *SimMsr*. We use a single point crossover with probability of P_{xover} to combine two different parents s_n^x and s_n^y. A mutated test path is replaced by a test path that is selected at random from the set of all possible test paths. A valid solution is a set of test cases in which there is no duplicate. The stopping criterion is 10,000 fitness evaluations, which is equal to the cost of 1,000 runs of ART with a candidate size 10 in terms of the resulting number of distance calculations. The pseudo-code of the employed GA is as follows:

(1) Sample a population G of m sets of test cases uniformly from the search space (i.e., the set of all possible valid sets with a given size n)
(2) Repeat until the stopping criterion is met
(3) Choose s_n^x and s_n^y from G
(4) $(\acute{s}_n^x, \acute{s}_n^y) := \text{crossover}\ (s_n^x, s_n^y, P_{xover})$
(5) Mutate $(\acute{s}_n^x, \acute{s}_n^y)$
(6) If valid $(\acute{s}_n^x, \acute{s}_n^y) \wedge$
$\min\ (f(\acute{s}_n^x), f(\acute{s}_n^y)) \leq \min\ (f(s_n^x), f(s_n^y))$
(7) Then $s_n^x := \acute{s}_n^x$ and $s_n^y := \acute{s}_n^y$
(8) Else If valid $(\acute{s}_n^y) \wedge$
$\min\ (f(s_n^x), f(\acute{s}_n^y)) \leq \min\ (f(s_n^x), f(s_n^y))$
(9) Then $s_n^y := \acute{s}_n^y$
(10) Else If valid $(\acute{s}_n^x) \wedge$
$\min\ (f(\acute{s}_n^x), f(s_n^y)) \leq \min\ (f(s_n^x), f(s_n^y))$
(11) Then $s_n^x := \acute{s}_n^x$

3. Impact of outliers and rank scaling solution

Unlike test suites for numerical applications, where the population of all possible input test data is distributed with similar density in the input space, it is not uncommon in MBT that a subset of test cases be very dissimilar to the rest of the test suite (outliers). For example, if the test suite is derived from a state machine and (1) the state machine contains a partition which is initiated by a transition from the initial state, (2) this partition has no transition to/from the rest of the state machine and (3) the triggers of the transitions in the segment are very different than the triggers of the transitions in the main part of the state machine, then the test suite generated from such a model will contain a small set of test cases, which will be very dissimilar to the rest of the test suite, that covers that segment. Investigating the behavior of STCS in such a situation is necessary in order to gain confidence about STCS effectiveness in the context of MBT. But because we are evaluating STCS on industrial case studies, and such artifacts are difficult to obtain in large numbers to support a systematic investigation, we perform simulations based on industrial case studies to increase realism.

We can show that both STCS_GA and STCS_ART will try to select half of the test cases from the outlier clusters (for simplicity, we will assume the presence of only one outlier cluster). The reason is that the similarity between any pair, in which one test case is from the main set of test cases and the other from the outlier set, would have a very low value compared to the other similarity values in the matrix. Therefore, to minimize $SimMsr(s_n)$, the selection algorithm is guided to select as many as possible of these pairs. Given a minimized test suite of size *n*, there will be *m* test cases from the main set, and *o* test cases from the outlier set, with the constraint $m+o=n$. The number of pairs in which the two test cases are from different sets would be $m*o$. Under the constraint $m+o=n$, the term $m*o$ is maximized when $m=o$, from which it follows $o=n/2$ (Schur-concave function [10]). Therefore, nearly half of test cases will be chosen from the outlier cluster regardless of the proportion of the cluster sizes. Consequently, if the outlier cluster is small and does not contain any fault revealing test case, the FDR of STCS will likely be low.

Since we expect poor results in the presence of no-fault revealing outlier, we suggest using a rank scaling technique to alleviate the problem. In this technique, the raw values in a similarity matrix are replaced by their rank. The rank is simply the index of the value after ordering all similarity values of the matrix in an array. This rank scaling approach is derived from solutions for solving outlier problem in statistics [11] and help decreases the large similarity differences between test cases from the outlier cluster and the rest of the test suite.

Notice that, in this report, we are assuming N (test suite size) small enough such that a $N*N$ matrix can be stored without significant overheads (this was the case for the two industrial case studies analyzed in this report). When this is not possible, and we need to compute the similarity values on the fly each time we evaluate the similarity of a set of test cases, a dynamic rank scaling is needed. For example, a data structure (e.g., a hash-table) could be used to store all the unique similarity values encountered so far during the search (e.g., while using STCS_GA). Rank scaling would hence be based on those values.

4. Empirical Study

In this section we report the design and results of our empirical analysis. The high-level goal of this study is to investigate under which circumstances, characterized by the correlation between similarities of test cases and their fault detection, and the distribution of test cases in their definition space, a STCS is most effective in terms of fault detection rate (FDR).

4.1. Test suites description

In this study, we test different hypotheses regarding the effectiveness of STCS on different input test suites to minimize. Given a test suite of size N, we can consider a $N*N$ matrix to represent the test suite in which all similarity pairs are stored (actually, only half of it is necessary, due to the symmetric property of the similarity functions).

These matrices are all based on the modification of test suites from two industrial case studies. However, we had to manipulate the matrices to create all the possible situations of a test suite with respect to the properties we want to investigate, as further explained below.

The SUT in case study A is a safety monitoring component in a safety-critical control system implemented in C++. A flattened version of the state machine representing the SUT consists of 70 states and 349 transitions. There are 15 real faults in the SUT which are detectable by a test suite automatically generated from a UML state machine representing the SUT's behaviour. The test suite, which is generated using our MBT tool (TRUST) [12], contains 281 abstract test cases (test paths) covering all round trip paths [13] in the state machine. Each test path either detects a certain fault or not regardless of its input data. In other word, the FDR values of the test paths of this case study are independent from input data.

The SUT in case study B is the core subsystem of a video-conference system which manages sending and receiving of multimedia streams implemented in C. As the previous case study, we deal with real faults detectable by an automatically generated test suite using TRUST. Case study B is smaller than A with 11 states, 70 transitions, 59 test cases (covering all round trip paths in the state machine) and only four detectable faults. But unlike case study A, the FDR of the test paths are not independent from input data. Depending on which data are chosen as input parameters for the events on the state machine, a fault may or may not be detected.

4.2. Research questions

The high level goal of this study leads to the following research questions:

RQ1. Under which conditions, with respect to the similarity of fault revealing test cases in a test suite, STCS performs best?

RQ1.1. Is STCS more effective if test cases which detect distinct faults are dissimilar?

RQ1.2. Is STCS more effective if test cases which detect common faults are similar?

These questions directly target the hypothesis on rewarding diversity, as discussed in Section 1, and seek to confirm it in the context of MBT. The diversity hypothesis is investigated with respect to two distinct properties through RQ1.1 and RQ1.2.

RQ2. Is rewarding diversity robust to small clusters of test case outliers (test cases which are very dissimilar to the rest of test suite)?

RQ3. What is the effect of rank scaling in the presence and absence of outliers?

RQ3.1. Does using rank scaled similarities improve STCS effectiveness in the presence of outliers?

RQ3.2. Does using rank scaled similarities impact negatively STCS effectiveness when there is no outlier?

The problem of outliers, discussed in Section 3, is being examined in RQ2. Our motivation, as mentioned above, is that in contrast to numerical applications, outliers are not a rare feature of test suites when they are generated using MBT. In question RQ3, the first sub-question RQ3.1 asks whether rank scaling is useful to alleviate the effect of outliers. RQ3.2 investigates whether rank scaling can reduce FDR when there is no outlier. If RQ3.2 shows that rank scaling does not reduce the FDR in such situations and RQ3.1 shows that rank scaling alleviates the outlier's problem, it is wise to always apply rank scaling in STCS. If results show that rank scaling does not reduce the FDR in such situations and that rank scaling alleviates the outlier problem, then it would be recommended to always apply rank scaling in STCS.

4.3. General settings of the experiments

We designed two experiments Exp1 to answer RQ1 and Exp2 to answer RQ2 and RQ3. In both experiments we use STCS_GA and STCS_ART based on trigger-guard encoding and NW as similarity function when it must be specified. Note that the results of our study in [3], which was based on only case study A, showed the same level of effectiveness for both NW and Jaccard Index. We had recommended the Jaccard Index in that study since it is easier to apply than NW. However, in case study B, NW provides much better results since the sequence of test path

elements matters regarding fault detection in this study and NW is a sequence-based function. Therefore, in this report NW is used for both case studies.

Since we have built this study based on our previous work, the overall settings of the algorithms are the same as our previous study settings. For GA, the stopping criterion is 10,000 fitness evaluations, the crossover probability is 0.75 and the population size is 50. For ART, the candidate size is 10 and 1000 repetitions (from which we select the best) are performed for each run of the algorithm to ensure fair comparisons with GA. More details regarding the settings and rationale behind our choices can be found in [2].

Each experiment uses input matrices which are generated by modifying similarity matrices of the case studies A and B, which we refer to as simulations. We repeat the experiments on different sample sizes (four for case study A and six for case study B) to check whether the results are consistent across the size range. Although the actual sizes for the sample sets are different for the two case studies, the percentage of selected test cases among all test suites is almost the same: sample sizes for experiments driven from case study A are equal to 5, 15, 25, and 35, and sample sizes for the experiments driven from case study B range from 3 to 8. The important point here is running the experiments on relatively small sample sizes, since this is the most interesting case in practice, when it is not possible to run many test cases on the actual hardware and platform (as for the industrial systems used as a case study in this report). Furthermore, for larger sizes all techniques converges to 100% FDR and differences will not be significant.

For both algorithms and all sample sizes, each experiment is repeated 1000 times (100 runs for search technique with different random seeds and 10 different input matrices per each matrix type to account for random variation in both search techniques and matrix generation). A rigorous statistical procedure has been used to evaluate and compare the effectiveness of these randomized algorithms [14].

4.4. Design and results of Exp1

To answer RQ1, we designed Exp1 where STCS_GA and STCS_ART, are applied on nine different types of input similarity matrices. These similarity matrices are artificially built—though based on case study A—to simulate all possible combinations of two properties of a test suite with respect to its test cases' similarities. Property X denotes the similarity between test cases that detect a common fault and Property Y denotes the similarity between test cases that detect distinct faults. In other words, RQ1.1 and RQ1.2 address the effect of Property Y and X on STCS effectiveness. In our simulations, each of these two properties can have three values: *High* (top 10%: [0.9,1.0]), *Low* (bottom 10%: [0.0,0.1]), and *Random* (randomly picked from the valid range: [0.0,1.0]), which makes nine possible combinations of the properties as an identifier for a test suite. For example, a test suite where test cases that detect a common fault are highly similar and those that detect different faults are very dissimilar, is identified by Property X=*High* and Property Y=*Low*. Note that, since the similarity functions we use only need to be positive and symmetric, when we generate matrices for our experiments, we do not need to validate each similarity value by checking its relationship with other values for other test case pairs in the same matrix.

To generate matrices with different property combinations while remaining as realistic as possible, we kept the original number of faults (15) and same failure rate as in case study A (74/281) and built matrices with sizes 300, 600, 6,000, and 12,000 (nine matrices for each matrix size). Recall that the reason for using different sizes is to test the independence of the results

from test suite size and therefore help the generalization of the results to larger case studies (i.e., does the technique scale?). Though this is only realistic when the system under test has already undergone significant verification, to make the analysis tractable, we assumed that each test case can find at most one fault. At this stage it is difficult to assess the consequences of this assumption and it therefore constitutes a threat to validity.

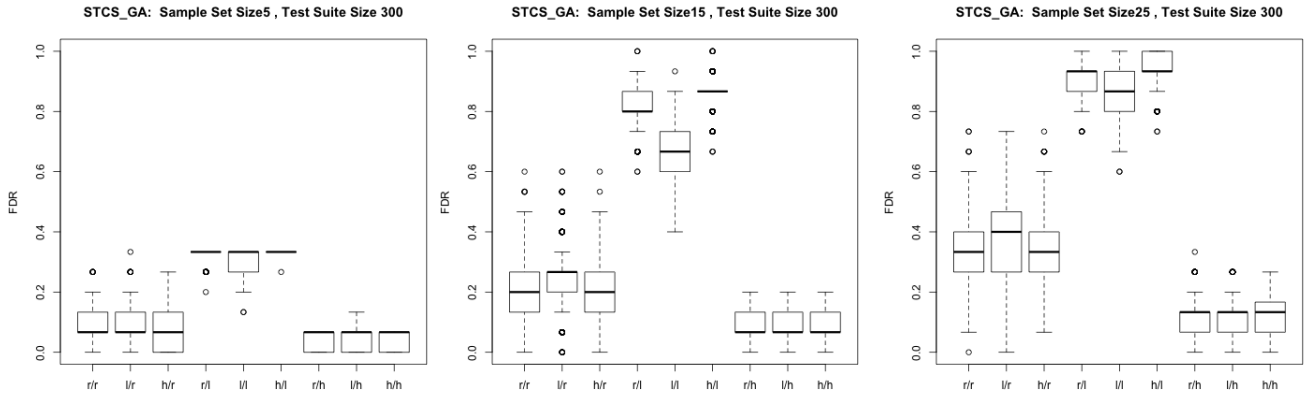
For each matrix type, 10 instances are generated. Both STCS_GA and STCS_ART are applied on these matrices 100 times, which yields a total of 1000 runs. In total, given that there are four sample sizes, nine matrix types, 1000 runs, and two selection techniques, then 288,000 (4*9*10*2*100*4) observations are collected in Exp1, each with an FDR value for the selected test cases. The FDR is the average number of faults detected by the selected test cases, for each run of the STCS, divided by the total number of faults (15).

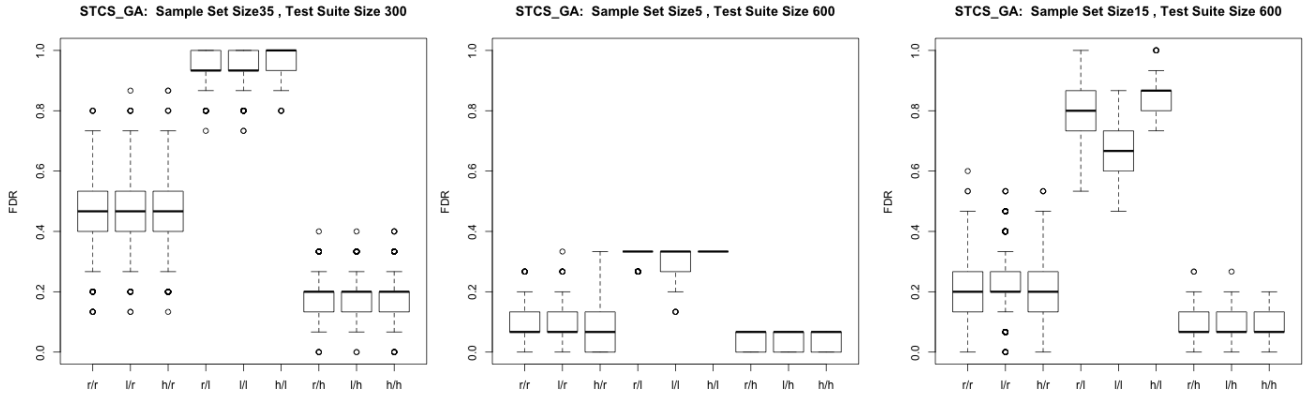
Figures 1 and 2 show results for Exp1 containing the FDR results (Figure 1) and the effect size (Figure 2) for four sample set sizes (5, 15, 25, and 35) and four matrix sizes (300, 600, 6000, and 12000) for each of the STCSs. The first observation is that, regardless of the type of SCTS, sample size, and matrix size, test suites with a *Low* value for Property Y show higher FDR. This means that the most important factor for ensuring the success of STCS is having test cases detecting distinct faults as far (dissimilar) as possible from each other. This confirms our hypothesis and answers RQ1.1.

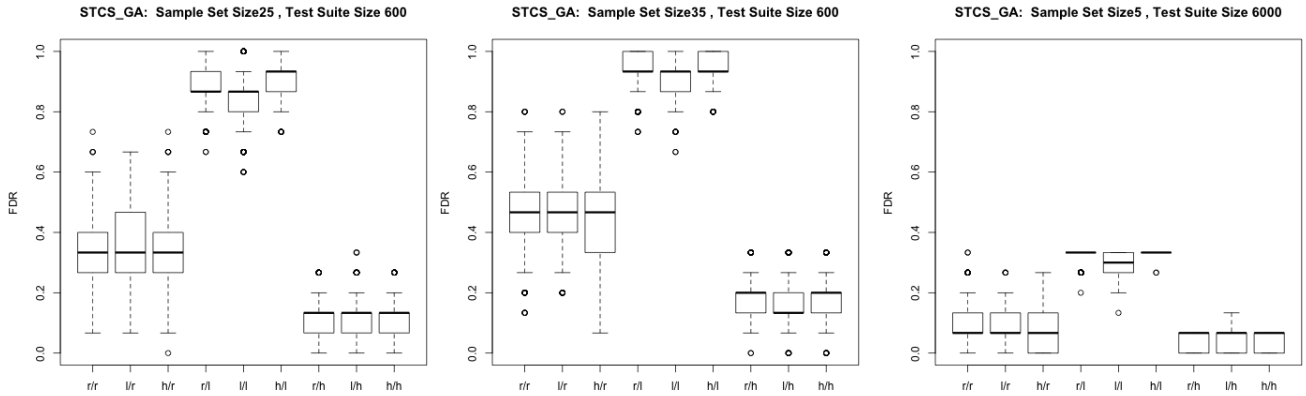
To answer RQ1.2, if we first look at cases where Property Y has a *Low* value, we can see significant differences in test suites with *High* values for Property X when compared to the others. This means that the combination of *High/Low* values for property X/Y is the best combination for STCS. This directly confirms the hypothesis discussed in RQ1. However, Property Y seems to have stronger effect since its value completely overrides the effect of Property X.

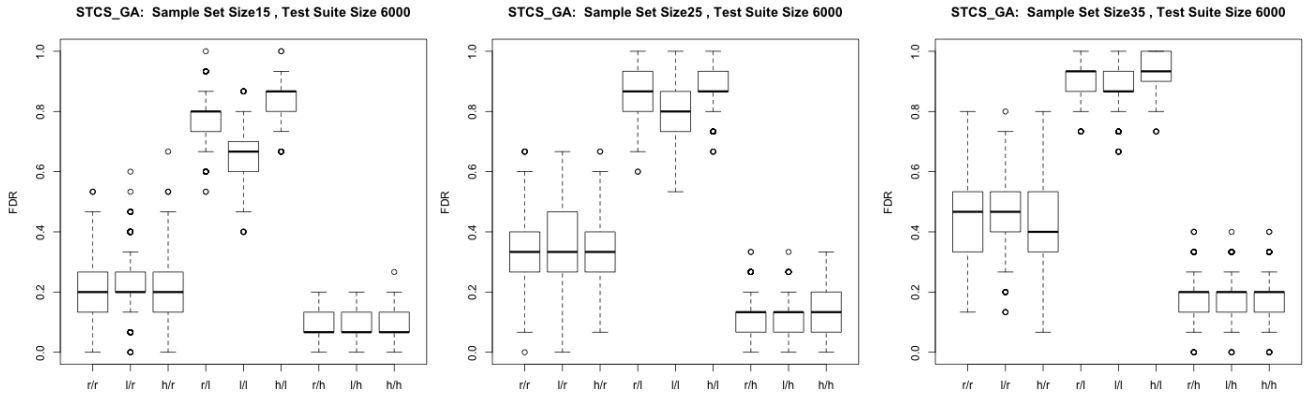
To gain more confidence in the conclusions drawn from this empirical study, we also carried out a series of statistical tests. For each of the 16 combinations of matrix sizes and test sample sizes, we used a Mann-Whitney U-test to compare the performance of the property combination *High/Low* against the other eight combinations. This test verifies whether two FDR distributions are statistically different. For STCS_GA, the p-values were always lower than our selected level of significance (0.05). For STCS_ART, resulting p-values were lower than 0.05 in all cases but four out of the 16*8=128 comparisons. This provides strong statistical evidence to support the claim that *High/Low* is the best condition under which to use STCS.

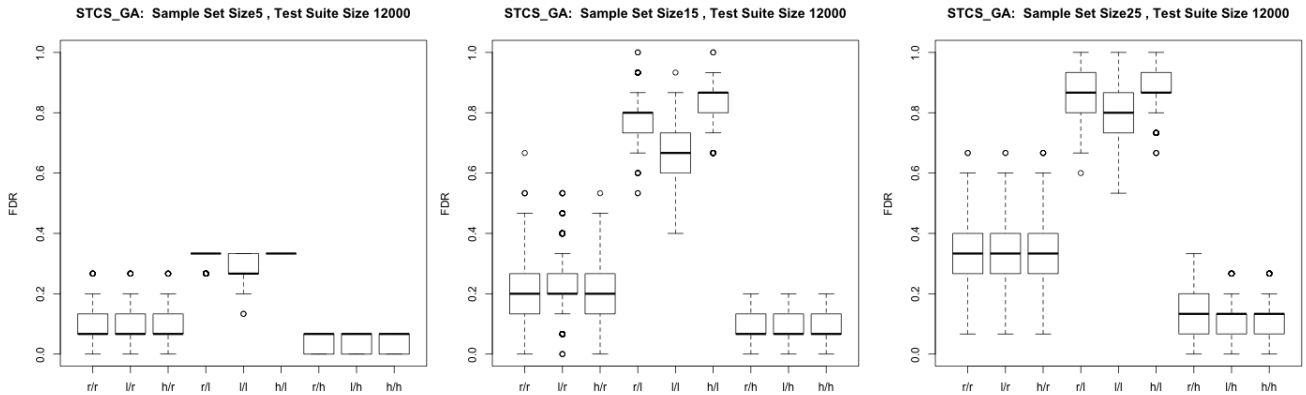
To quantify the magnitude of improvement in a standardized way, in Figure 2 we plot the effect size measure of STCS_GA and STCS_ART for different sample and matrix sizes using the Vargha-Delaney's *A* statistic. This statistic estimates the probability that a data point randomly taken from a set (i.e., a probability distribution) will have higher value than another point randomly taken from a second data set. When the two distributions are the same, we would have *A=0.5*. The results in Figure 2 show that, most of the time, the *A* values are close to 1. This means that, for the *High/Low* combination, it is nearly certain that STCS will yield better results than in the other eight cases, even when we take into account the variance of the results due to the randomized nature of these algorithms.

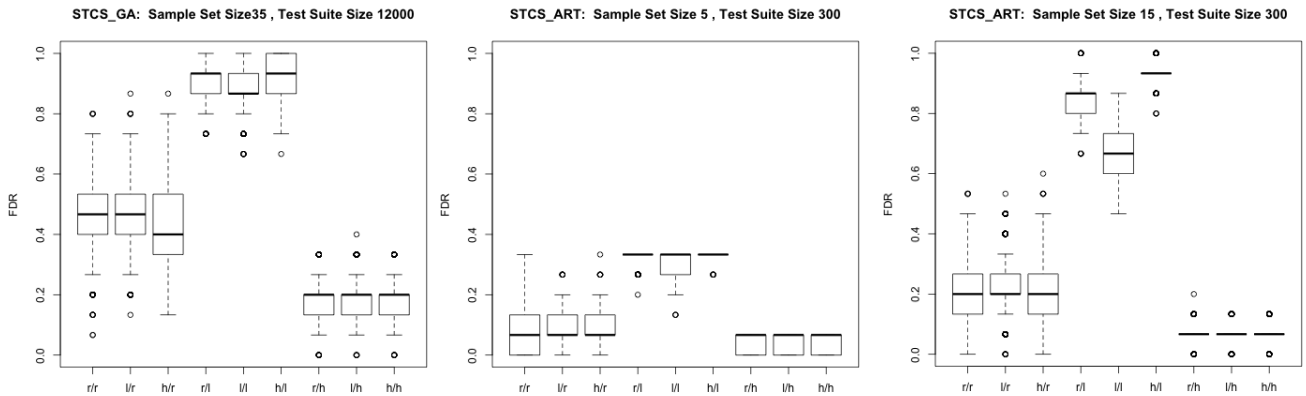

STCS_GA: Sample Set Size5 , Test Suite Size 300
STCS_GA: Sample Set Size15 , Test Suite Size 300
STCS_GA: Sample Set Size25 , Test Suite Size 300
FDR
0.0
0.2
0.4
0.6
0.8
1.0
r/r
l/r
h/r
r/l
l/l
h/l
r/h
l/h
h/h

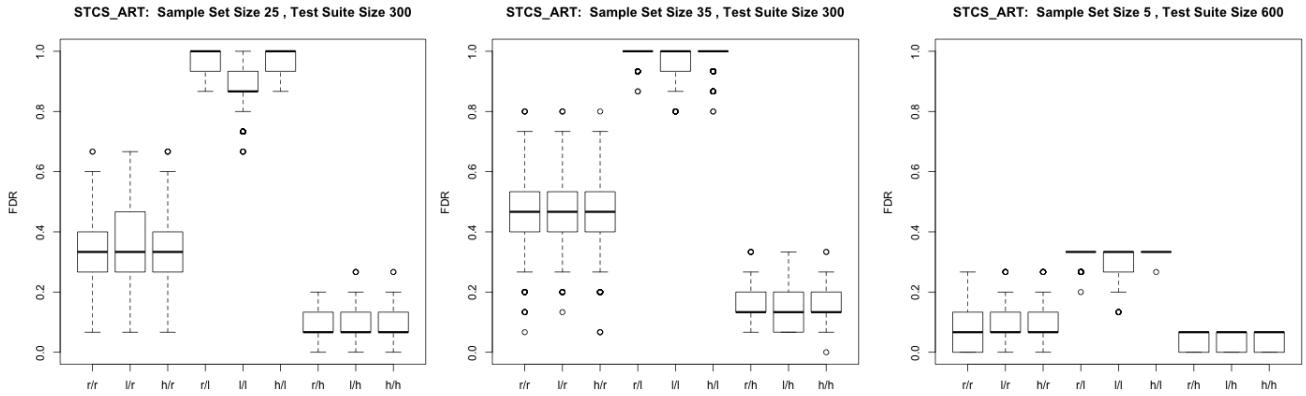

STCS_GA: Sample Set Size35 , Test Suite Size 300
STCS_GA: Sample Set Size5 , Test Suite Size 600
STCS_GA: Sample Set Size15 , Test Suite Size 600
FDR
0.0
0.2
0.4
0.6
0.8
1.0
r/r
l/r
h/r
r/l
l/l
h/l
r/h
l/h
h/h

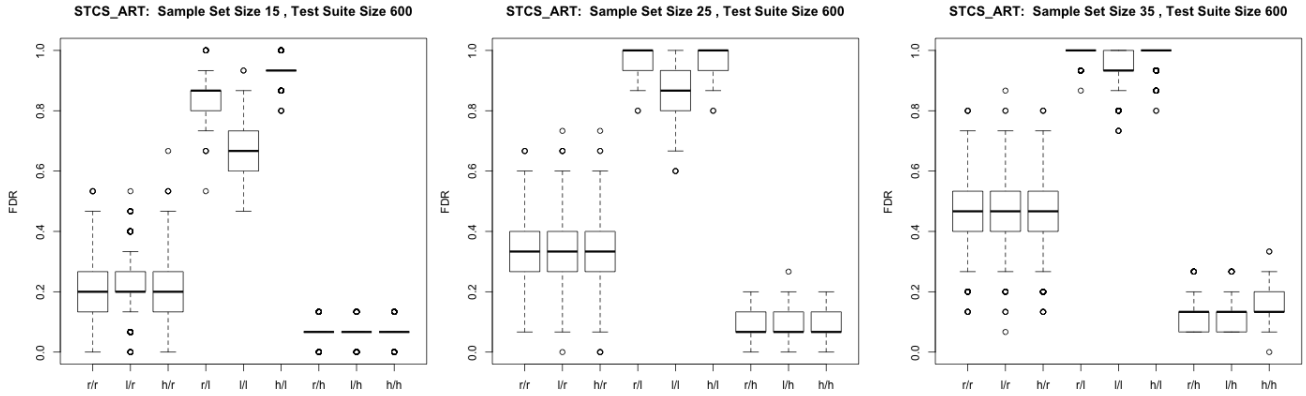

STCS_GA: Sample Set Size25 , Test Suite Size 600
STCS_GA: Sample Set Size35 , Test Suite Size 600
STCS_GA: Sample Set Size5 , Test Suite Size 6000
FDR
0.0
0.2
0.4
0.6
0.8
1.0
r/r
l/r
h/r
r/l
l/l
h/l
r/h
l/h
h/h

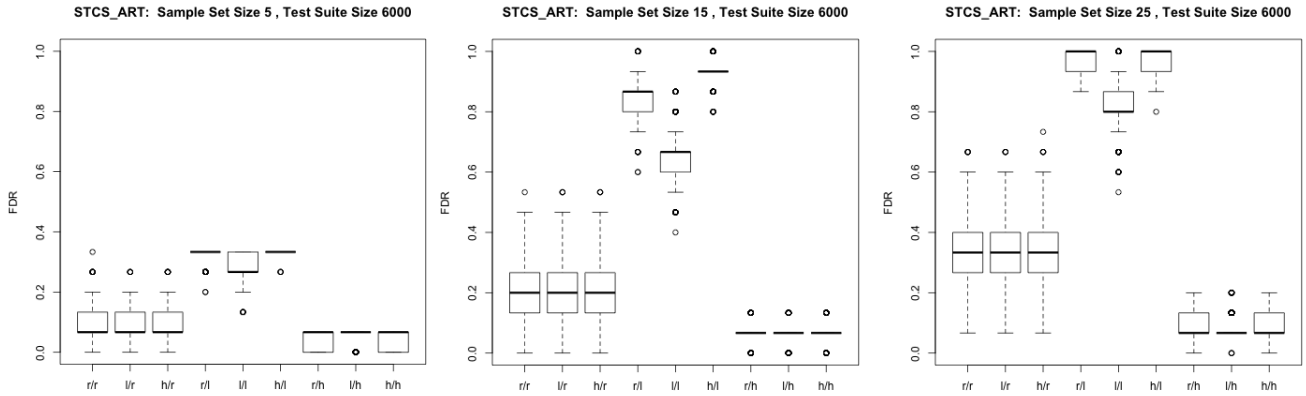

STCS_GA: Sample Set Size15 , Test Suite Size 6000
STCS_GA: Sample Set Size25 , Test Suite Size 6000
STCS_GA: Sample Set Size35 , Test Suite Size 6000
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h


STCS_GA: Sample Set Size5 , Test Suite Size 12000
STCS_GA: Sample Set Size15 , Test Suite Size 12000
STCS_GA: Sample Set Size25 , Test Suite Size 12000
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h


STCS_GA: Sample Set Size35 , Test Suite Size 12000
STCS_ART: Sample Set Size 5 , Test Suite Size 300
STCS_ART: Sample Set Size 15 , Test Suite Size 300
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h


STCS_ART: Sample Set Size 25 , Test Suite Size 300
STCS_ART: Sample Set Size 35 , Test Suite Size 300
STCS_ART: Sample Set Size 5 , Test Suite Size 600
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h


STCS_ART: Sample Set Size 15 , Test Suite Size 600
STCS_ART: Sample Set Size 25 , Test Suite Size 600
STCS_ART: Sample Set Size 35 , Test Suite Size 600
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h


STCS_ART: Sample Set Size 5 , Test Suite Size 6000
STCS_ART: Sample Set Size 15 , Test Suite Size 6000
STCS_ART: Sample Set Size 25 , Test Suite Size 6000
FDR
r/r l/r h/r r/l l/l h/l r/h l/h h/h

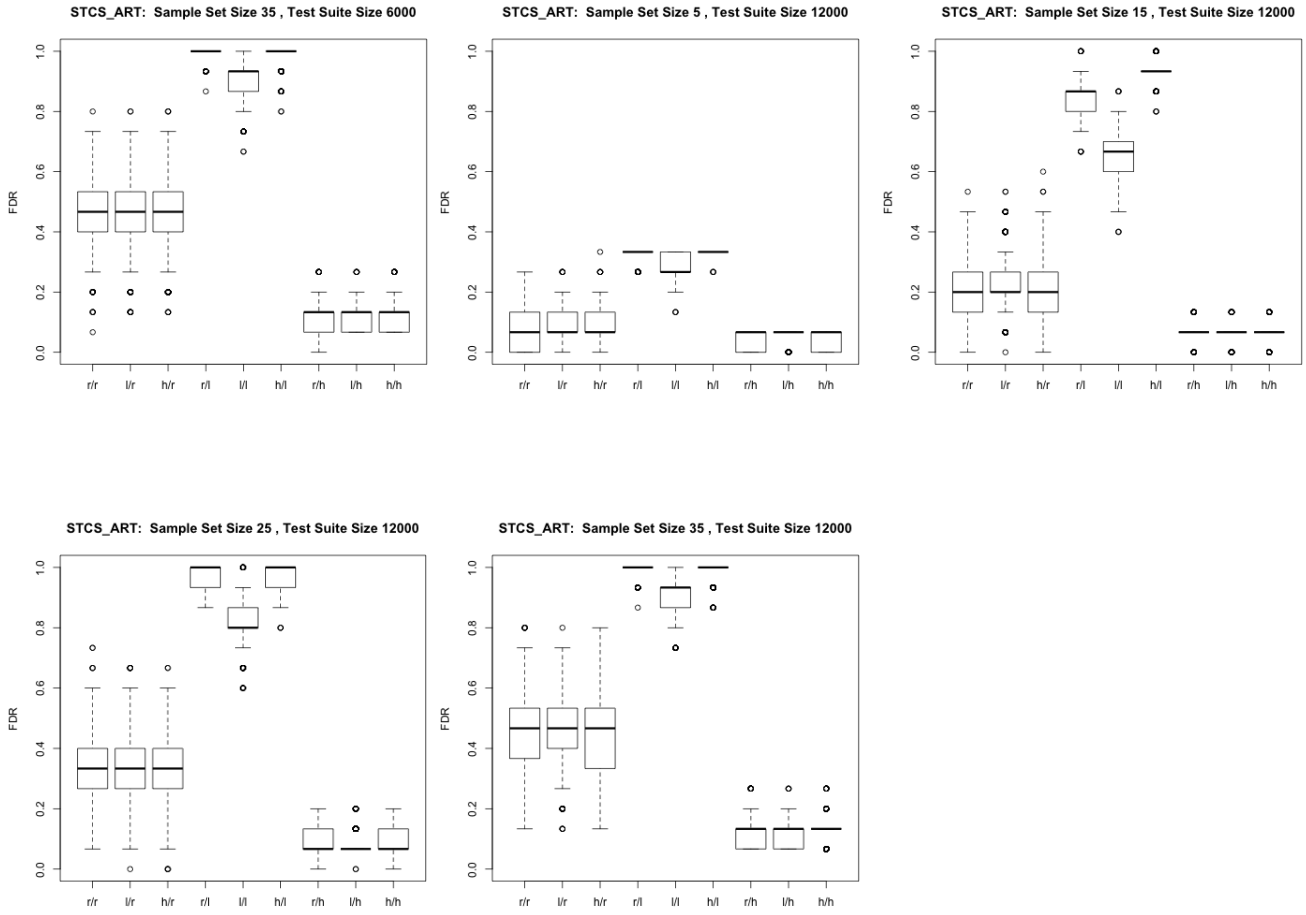
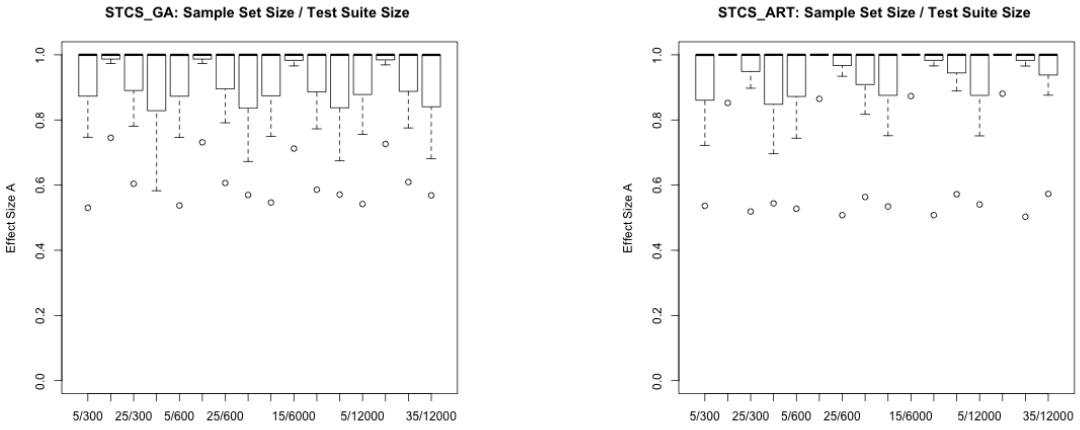
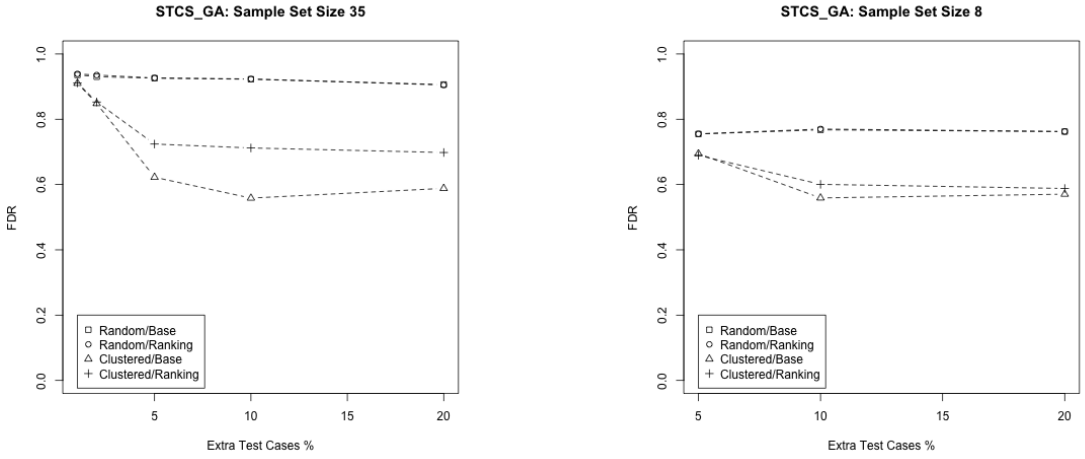


Figure 1. FDR of a sample set (of size 5, 15, 25, and 35) of test cases selected by STCS_GA and STCS_ART from different matrix types (of size 300, 600, 6000, and 12000). Matrix types on X_Axis are identified as Property X/Property Y where each property can be random (r), low (l) or high (h). Each boxplot shows 1000 observations (100 STCS runs per matrix on 10 different matrix instances).



(a) (b)

Figure 2. Effect size measure A (each calculated out of 1000 observations) for FDR of a sample set selected by STCS_GA (a) and STCS_ART (b) shown as boxplots for the eight comparisons. The effect size compares the High/Low matrix type with the all other eight matrix types of Figure 1. X_Axis shows the sample size/matrix size.



(a) (b)

Figure 3. FDR of a sample set selected by STCS_GA from test suites based on case study A with size 35 (a) and case study B with size 8 (b). Four combinations are compared: with (Clustered) or without outliers (Random), and using rank scaling (Ranking) or not (Base). The graphs show the average FDR over 1000 STCS_GA runs. X_Axis shows the percentage of outliers.

4.5. Design and results of Exp2

For Exp2, we apply STCS_GA and STCS_ART on four types of matrices per case study. We manipulated the original matrices from each case study to examine the effect of outliers on the FDR of the test suites. We did so by adding extra percentages of outlier test cases. For case study A and B, respectively, we built matrices with 1, 2, 5, 10, and 20 and 5, 10, and 20 percent extra test cases (1 and 2 percent would not make sense for the smaller case study B with only 59 test cases). Four types of matrices are generated for each case study and size: (1) *Random/Base*: The original matrix from the case study plus extra test cases with random similarity values in the same range of similarity values as in the original matrix. This matrix is the baseline for the FDR comparisons; (2) *Cluster/Base*: The original matrix plus extra test cases with random similarity to each other but very low similarity (outliers) to the rest of the test suite (original test cases). This low similarity value must be set to be much lower than the minimum values within each of the groups containing the original and additional test cases. If *min* and *max* are the minimum and maximum values in the original matrix, we first change the matrix by replacing every value *x* with *x*+10*(*max-min*) to ensure much higher NW similarity values among the original test cases compared to such values with outliers. The NW values between outliers are then generated to be in the same range as the original matrix. Last, to simulate a low similarity between the outliers and the original test cases, we set the NW value between them to zero. The constructed matrix therefore represents the situation where outlier test paths are present in the test suite; (3) *Random/Ranking*: The same matrix as *Random/Base* but after applying rank scaling as introduced in the research question subsection; (4) *Cluster/Ranking*: The same matrix as *Cluster/Base* but after applying rank scaling.

To answer RQ2, we compare the FDR of a selected subset of test cases (for four different sizes) from a test suite represented by the *Cluster/Base* matrix with the FDR of a same size subset using the *Random/Base* matrix. This comparison investigates the effect of outliers on the STCS effectiveness.

To investigate RQ3, we compare STCS effectiveness on the matrices from *Cluster/Ranking* and *Cluster/base*, we will assess whether rank scaling has significantly alleviated the effect of the outliers (RQ3.1). We also compare the effectiveness of STCS on the *Random/Ranking* and *Random/Base* matrices to check for possible negative effects of rank scaling when there is no outlier (RQ3.2).

We generate 10 instances of each of the 32 matrices (four matrix types and eight outlier percentages in the two case studies) to account for random variation in matrix generation. Both STCS_GA and STCS_ART are applied on these matrices 100 times to account for random variation in search techniques. In total, given that there are four sample sizes in case study A and six sample sizes in case study B, 320 matrices, 100 runs, and two selection techniques, then 640,000 (10*320*100*2) observations are collected for Exp2. Each observation has an FDR value for the selected test cases. The FDR calculation for case study A is the same as for Exp1 but is different for case study B, since, in the latter case, whether each test path detects a fault depends on which input data is used. For case study B, we randomly (with equal probability for each input data value) generated 10 different test cases per test path. Therefore, probability P_f of finding a specific fault f with the selected subset of test paths is equal to one minus the probability of not finding the fault by any of the test paths in the chosen set: $P_f = (1 - \prod_{i=1}^{n}(1 - p_i))$ where n is the size of the subset and p_i is the estimated probability of detecting fault f with test path i in the subset: number of times the fault is detected by the 10 test cases generated for that test path divided by 10. The FDR is hence computed by averaging these probabilities $\sum P_f / |F|$, where $|F|$ is the number of faults. From the results of Exp2, answering RQ2 and RQ3, Figure 3 and Figure 4 are chosen to show one representative example since the trend is the same over different sample sizes and algorithms for case study A. In Figure 3.a, the clear gap between *Cluster/Base* and *Random/Base* shows a strong drop in STCS effectiveness in the presence of outliers (RQ2). Comparing *Cluster/Base* and *Cluster/Ranking* we can clearly see that rank scaling helps STCS improve its

effectiveness in the presence of outliers (RQ3.1) and comparing *Random/Base* and *Random/Ranking* clearly shows there is no reduction in FDR when there is no outlier in the test suite (RQ3.1).

In case study B, outliers also decrease effectiveness of STCS, though to a lesser extent (RQ2), and rank scaling once again does not compromise the potential FDR for test suites without outliers (RQ3.2). However, as it can be seen in the Figure 3.b, the improvement for case study B when comparing *Cluster/Base* and *Cluster/Ranking* is relatively small (RQ3.1), perhaps in part because the impact of outliers is already smaller to start with in this case study.

As in the previous experiment, to get more reliable results, we also carried out a rigorous statistical procedure using Mann-Whitney U-tests and Vargha- Delaney's *A* statistics (effect size) to compare FDR distributions across the four types of matrices. Comparing the performance of *Random/Base* with *Random/Ranking* (RQ3.2) yields p-values lower than 0.05 in only one case out of 20 comparisons (five extra test case percentages time four matrices) for STCS_GA and two out of 20 comparisons for STCS_ART, where, even in those cases, the FDR difference between *Random/Base* and *Random/Ranking* is practically negligible. This statistically confirms that rank scaling is not particularly harmful in most cases when no outlier is present. However, when we compare *Cluster/Base* against *Cluster/Ranking* (RQ3.1), we obtain 11 cases with significant p-values for STCS_GA, and six cases for STCS_ART.

In Figure 4, we plot the effect size measure of STCS_GA for different sample and matrix sizes when we compare *Cluster/Base* against *Cluster/Ranking* (RQ3.1) in case study A. For small sample sizes and small outlier cluster, the effect is minimal (i.e., very close to 0.5). However, for larger sizes, the effect gets much stronger (close to 0.7).

As explained before, the main reason for which we apply rank scaling is to balance the distribution of the selected test cases from each cluster of outliers (if present). To examine this phenomenon, we considered one scenario (case study A with 20% extra test cases forming a cluster of 56 test case outliers) and applied STCS_GA for selecting test case subsets (four sample sizes). Table 1 shows the average number of test cases taken from the outlier cluster with and without rank scaling. The best column shows the optimal number of test cases if we would select by only considering the size of the test suite and its outlier cluster, as expressed by the formula below.

$$Best = 1 + [\frac{clusterSize * (sampleSize - 1)}{testSuiteSize}]$$

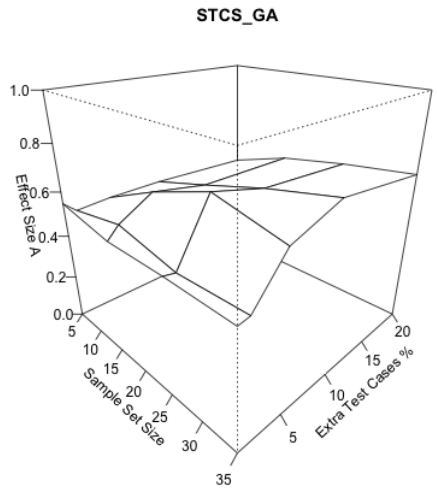


Figure 4. The effect size measure A for FDR of sample sets selected by STCS_GA from the test suite driven from case study A. X and Y axes show the outliers percentage and the sample set size.

Table 1. Average number of test cases selected by STCS_GA from the outlier cluster (20% extra test cases on the case study A) with and without rank scaling.

Sample Size	Best	No Ranking	Ranking
5	1	2.95	2.94
15	3	7.06	6.77
25	4	12.01	10.61
35	6	17.00	14.41

Based on the results in Table 1—Note the relation between Table 1 and Figure 3: the last row of Table 1 corresponds to Figure 3.a, with 20% extra test cases — it is clear that without rank scaling roughly half of the sample set is taken from the outlier cluster. The data suggest that rank scaling partially alleviates the problem for larger sample sizes. We get better improvement for larger sample sizes and the reason why this is the case will require further investigation.

One possible alternative to rank scaling for solving the outlier problem could be an approach that can be summarized as (1) finding the outlier cluster(s) using a clustering technique and identifying an outlier cluster based on the ratio of the inter-cluster distances to the intra-cluster distances (2) assigning a sample size to the outlier cluster based on the proportion of its size to the entire test suite size (3) and finally applying the STCS separately on the outlier and the main test cases. In previous work [2], we found that clustering techniques were less effective than STCS. Furthermore, rank scaling is easier and computationally cheaper than clustering techniques. However, hybrid combinations would be promising areas for further research.

4.6. Discussion on threats to validity of the results

This study was conducted according to recently proposed guidelines for conducting empirical studies in search-based testing [15] and using statistical tests to assess randomized algorithms in software engineering [14]. Regarding *construct validity* of the experiments, the most important factor is the validity of the measures used for assessing FDR and similarity comparisons. These measures are taken from previously published studies [2-4] and their validity are already discussed there. Another remaining concern is the artificially generated similarity values in the experiments. As discussed in the background section, we are using the NW similarity measure, which entails no constraint on the different pairs of similarities. Therefore, the assignment of *High*, *Low*, and *Random* values cannot lead to incorrect matrices. However, the assumption in Exp1 that each test case can find at most one fault constitutes a threat to validity of the results. A more general experiment where each test case can find each fault with a certain probability should be conducted to achieve more reliable results.

The randomized nature of the employed algorithms poses a threat to *internal validity*. To account for it, the experiments were run many times with different random seeds, thus leading 1000 observations for each case study/sample size/search technique/matrix type combination (100 runs of search technique on 10 randomly generated input matrices). In addition, a rigorous statistical procedure (comprising significance tests and effect size measures) has been used to strengthen the *conclusion validity* of the results.

To cope with *external validity*, we conducted experiments using many different combinations of sample sizes, test suite sizes, case studies, and STCS techniques. In particular, the use of two industrial systems to drive the simulations (by retaining some of their characteristics such as failure rate of test cases and number of faults) provides stronger support to the applicability of our approach to other industrial systems. But, as for all empirical studies, our results might not generalize to other case studies and only replications will help build confidence.

5. Related work

STCS for MBT was first introduced in [16], where sequences of transitions in a Labeled Transition System model of the SUT are used for representing test paths. The similarity function is simply counting the common transitions in two test paths and a Greedy Search is used for minimizing the sum of pair similarities. Later, Hemmati *et al.* [4] introduced and improved STCS for UML based testing by using a trigger-guard based encoding of test paths, by using better similarity measures [3] and by resorting to more powerful search techniques [2].

Except for these works on model-based STCS, diversifying test cases has been studied on code-based test case selection, minimization and prioritization, mostly in the context of regression testing. The basis for computing test case similarity in these studies is usually on code coverage or on some other execution information. For example, in [17] and in [18], all def-use pairs coverage and a sequence of memory operations are used to calculate the similarities, respectively.

To the best of authors' knowledge, no existing study systematically investigates the impact of test suite properties on STCS in the context of MBT. Similar studies published to date have been conducted in the numerical application domain to examine the effect of test suite properties, with respect to test case similarities and their fault detection, on the ART algorithm. Several papers have been published on this subject [1], in which for example optimal conditions for ART have been theoretically studied [19]. However, as discussed in Section 3, MBT is very different from the unit-testing of numerical applications in terms of the distribution of input test data in the input space (e.g., clusters of outliers are unlikely in the numerical application domain).

6. Conclusion and future work

In previous studies we proposed similarity-based test case selection (STCS) techniques to reduce the cost of model-based testing (MBT) [2-4]. Though the technique was successfully applied on one industrial system, we needed more empirical evidence to support the idea that maximizing the diversity of test cases was a good principle for test case selection and understand under which conditions.

In this report, we conducted a large scale simulation, based on two industrial case studies, to investigate, in a controlled manner, how relevant properties of a test suite affect the effectiveness of STCS. When considering properties are about the relationship between fault detection and similarity distributions among test cases, our results showed that the most ideal situation for a STCS is when, in a test suite, (1) test cases that detect a common fault are similar and (2) test cases which detect distinct faults are dissimilar. Our empirical study shows that property (2) is much more important than property (1). This result will help us devise improved similarity functions in the future, which in turn will result into more effective STCS.

In this report, we also investigated the problem of outliers in a test suite—which are not unlikely to happen in MBT—that could compromise the performance of STCS. Results confirmed the significant impact of outliers and an approach, based on using rank scaling measurement instead of raw similarity values, was proposed to address the outlier problem. Though rank scaling had a positive effect, it only partially addressed the outlier problem and additional strategies remain to investigate.

Future work will examine other solutions for the outlier problem based on combining clustering and STCS techniques. We will also use the insights that we gained from this study to develop techniques to improve STCS.

7. Acknowledgement

The authors wish to thank Marius Liaaen, from Tandberg AS, now part of Cisco, for helping us in conducting experiments.

8. References

[1] T. Y. Chen, F.-C. Kuoa, R. G. Merkela, and T. H. Tseb, "Adaptive Random Testing: The ART of test case diversity," *Journal of Systems and Software*, vol. 83, pp. 60-66, 2010.

[2] H. Hemmati, A. Arcuri, and L. Briand, "Reducing the Cost of Model-Based Testing through Test Case Diversity," in *22nd IFIP International conference on Testing Software and Systems (ICTSS), formerly TestCom/FATES*, 2010.

[3] H. Hemmati and L. Briand, "An Industrial Investigation of Similarity Measures for Model-Based Test Case Selection," in *21st IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2010.

[4] H. Hemmati, L. Briand, A. Arcuri, and S. Ali, "An Enhanced Test Case Selection Approach for Model-Based Testing: An Industrial Case Study," in *18th ACM International Symposium on Foundations of Software Engineering (FSE)*, 2010.

[5] P. N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*: Addison Wesley, 2006.

[6] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison, *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*: Cambridge University Press, 1999.

[7] R. Xu and D. C. Wunsch II, "Survey of Clustering Algorithms," *IEEE Transactions on Neural Netwoks*, vol. 16, pp. 645-678, 2005.

[8] G. Dong and J. Pei, *Sequence Data Mining*: springer, 2007.

[9] A. P. Mathur, *Foundations of Software Testing*, 1 ed.: Addison-Wesley Professional, 2008.
[10] P. J. Boland, H. Singh, and B. Cukic, "Comparing partition and random testing via majorization and Schur functions," *IEEE Transactions on Software Engineering*, vol. 29, pp. 88-94, 2003.
[11] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, 3 ed.: Chapman & Hall, 2003.
[12] S. Ali, H. Hemmati, N. E. Holt, E. Arisholm, and L. Briand, "Model Transformations as a Strategy to Automate Model-Based Testing - A Tool and Industrial Case Studies," Simula Research Laboratory, Technical Report(2010-01)2010.
[13] R. V. Binder, *Testing Object-Oriented Systems: Models, Patterns, and Tools*: Addison-Wesley Professional, 1999.
[14] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," Simula Research Laboratory (2010-10) 2010.
[15] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A Systematic Review of the Application and Empirical Investigation of Search-based Test-Case Generation," *IEEE Transactions on Software Engineering, Special issue on Search-Based Software Engineering (SBSE), in press*, 2010.
[16] E. G. Cartaxo, P. D. L. Machado, and F. G. O. Neto, "On the use of a similarity function for test case selection in the context of model-based testing," *Software Testing, Verification and Reliability*, 2009.
[17] A. d. S. Simão, R. F. d. Mello, and L. J. Senger, "A Technique to Reduce the Test Case Suites for Regression Testing Based on a Self-Organizing Neural Network Architecture," in *30th Annual International Computer Software and Applications Conference (COMPSAC)*, 2006.
[18] M. K. Ramanathan, M. Koyutürk, A. Grama, and S. Jagannathan, "PHALANX: a graph-theoretic framework for test case prioritization," in *23rd Annual ACM Symposium on Applied Computing*, 2008.
[19] T. Y. Chen and R. Merkel, "An upper bound on software testing effectiveness," *ACM Transactions on Software Engineering and Methodology*, vol. 17, 2008.