Enhanced Adaptive Random Test Case Prioritization for Model-based Test Suites

Tomas Pospisil, Jan Sobotka and Jiri Novak

Czech Technical University in Prague, Faculty of Electrical Engineering, Technicka 2, 166 27 Prague, Czech Republic, E-mail: pospito7@fel.cvut.cz, jan.sobotka@fel.cvut.cz, jnovak@fel.cvut.cz

Abstract: Adaptive Random Prioritization is a Test Case Prioritization technique which orders test cases within a test suite with a goal of earlier fault detection using semi-random heuristics. Compared to other Test Case Prioritization methods, Adaptive Random Prioritization has only, an "average fault detection performance. However, it is less sensitive to some test suite features which negatively affect fault detection performance than other TCP techniques due to its semi-random nature. The article proposes an improved version of Adaptive Random Prioritization technique. The key idea behind the presented enhancement is to extend the test case selection process with additional information about control flow and change of test statements coverage, of a test suite. The enhancement replaces the original Test set distance function with a Multi-Criteria Decision-Making method. Validity of the proposed method is evaluated on data from six embedded systems. The evaluation criterion is fault detection performance expressed by Average Percentage of Faults Detection metric and Â12 statistic. The proposed improvement achieved better fault detection performance for all of the examined systems.

Keywords: Adaptive Random Testing; Model-based testing; Multi-Criteria Decision-Making

1 Introduction

Testing of modern electronic/software systems is an essential activity in the quality assurance process. Many approaches for how to design and execute a test suite currently exist. A test suite consists of individual test cases. Since testing resources are always limited, a research of optimization techniques focused on cost reduction is needed [1]. One way of testing automation is Model-Based Testing (MBT) [2, 3]. In MBT world, test cases are generated by a software tool from a model. E.g., System-under-Test (SUT) behavior is modeled by Finite State Machine and a test is an oriented path through the automaton. A test suite is produced by a graph traversal algorithm like a Breadth-First search. Due to the automatic nature, a Model-Based Test Suite can contain a tremendous number of

generated test cases. Optimization by ordering of these test cases according to given (structural model coverage, random and stochastic, data coverage ...) criteria is required. This problem is dealt with a category of techniques referred to as Test Case Prioritization (TCP) [4].

Recently, TCP techniques were explored mainly in code-based and regression testing areas [5, 6]. On the other hand, TCP techniques for MBT approach are not so well investigated. The main difference between these areas is that general TCP techniques in MBT do not use any external information, as historical information [6] or expert knowledge (Risk-based TCP [7]). Since this context is not so profoundly explored, further research in this area is needed.

Results of a study on TCP techniques in the MBT area [8] show that no TCP technique has superior performance. Examined techniques performance varies for different scenarios. The study indicates TCP techniques based on Path Complexity and additional coverage of statements achieve good overall performance, but they are affected by the size of the test cases that fail. In contrast, Adaptive Random Prioritization (ARP) is less sensitive to the size of the test cases that fail, but they have only moderate fault detection results.

Based on these results, an enhanced ARP technique is proposed. The presented technique replaces a standard *Test set distance* function (see Section 2.1) with a Multi-Criteria Decision-Making (MCDM) method [9]. The method combines criteria based not only on the distance metric but also on other test case features. These features include additional information to TCP process from various sources, which can be code-based, model-based, can use previous tests results, etc. The presented technique is mainly designed for testing of the embedded systems, described with high-level behavioral models and for cases where other information as source code, or fault history is not available. Therefore, the proposed technique uses only the MBT metrics with high fault detection performance. The primary goal is to improve the fault detection performance of adaptive random based techniques and simultaneously preserve their low sensitivity to the size of the test cases that fail.

2 Background

This section contains a summary of the ARP technique and other methods that are incorporated in proposed MCDM ARP enhancement, specifically Path Complexity and Additional coverage techniques. Besides that, a short overview of TCP in MBT area is outlined at the end of the section.

2.1 Adaptive Random Prioritization

Adaptive Random Prioritization technique [10] belongs to a family of Adaptive Random Strategy (ARS) techniques. Chen et al. [11] initially proposed the ARS as an online test case generation method for software with numerical inputs. ARS was proposed as an alternative to a pure random test input generation strategy. The idea behind ARS is an input space estimation that selects areas that cause failures and spreads the test cases more efficiently over the input space.

The ARP is an application of ARS to the TCP problem. The technique is described in the Algorithm – Part 1. Where the *UTS* input is an unsorted test suite, and the output of the function is a prioritized test suite (*PTS*). In the first step (line 3), the algorithm randomly selects a test case. It is saved as the first prioritized test case in *PTS* and subsequently removed from the *UTS* set (lines 4 and 5).

Algorithm – Part 1: The main procedure

```
1: function Prioritize(UTS)
2:         PTS ← Ø
3:         first ← randomChoice(UTS)
4:         PTS.add(first)
5:         UTS.remove(first)
6:         while UTS ≠ Ø do
7:                 cand ← genCandSet(UTS)
8:                 nextTC ← selectNext(PTS, cand)
9:                 PTS.add(nextTC)
10:                UTS.remove(nextTC)
11:        end while
12: return PTS
13: end
```

The prioritization of the suite is executed within the main loop on lines 6 to 11. In the first part of the loop (line 7), a set of candidates is generated according to Part 2 of the Algorithm.

Algorithm – Part 2: Candidates generation

- *UTS* is the test suite
- *cand* is the candidate set
- *candMax* is the maximal size of *cand* set
- *S*: {s_1, s_2, ...} is set of statements
- *S'*: {s'_1, s'_2, ...} is set of statements

```
1: function genCandSet (UTS)
2:         S ← Ø
3:         S' ← Ø
4:         cand ← Ø
5:         TC ← randomSelect(UTS)
6:         S ← statements covered by TC
```

```
7:      if S' ∪ S = S' then return cand
8:      S'.add(S)
9:      cand.add(TC)
10:     if cand.size == candMax then return cand
11:     goto 5
12: end
```

The candidate set *cand* is iteratively selected by the candidate generation function. In each iteration step, the generation process randomly selects a not-yet-prioritized test case (Part 2 – line 5). The selection process continues until newly selected test cases increase coverage of candidate set (Part 2 – line 5 to 10), or until the maximum number of candidates is reached.

The next step (Part 1 – Line 8) is the *selectNextTestCase* function, which selects a test case from the candidate set; Part 3 describes this function in detail. The *selectNextTestCase* function is based on functions f_1 and f_2. Function f_1 (Part 3 – line 5) is *Test case distance* function that calculates distances between candidates and the test cases that were already prioritized and saves them into the distance matrix *d*. Jiang et al. [10] used Jaccard distance function [12] instead of Euclidean distance function, which was proposed for the original ARS. Zhou [13] proposed an application of modified Manhattan distance function. Lately, Zhou et al. compared both distance functions in an empirical study [14]. The study shows that the Manhattan function provides better fault detection performance than Jaccard in the code-based context. Further improvement was proposed by Coutinho et al. [15], where Similarity function was implemented instead of distance functions.

Algorithm – Part 3: Next test case selection

- *PTS* is the already prioritized test sequence
- *cand* is the candidate set

```
1: function selectNextTestCase(PTS, cand)
2:      d ← array[PTS.size][cand.size]
3:      for i = 0 to PTS.size - 1 do
4:          for j = 0 to cand.size - 1 do
5:              d[i,j] ← f1(PTS[i], cand [j])
6:          end for
7:      end for
8:      index ← f2(d)
9:      nextTestCase ← cand.get(index)
10: return nextTestCase
11: end
```

Function f_2 is *Test set distance* function that returns the index of the selected test case that is farthest away from the prioritized set (Part 3 – line 8). This function can select test cases according to several prioritization rules. For example, the rule MaxMin – maximum of the minimum distances (similarities) first, determines the smallest distance of each candidate to all already prioritized test cases and then

finds a candidate with the lagest value of those minimal distances. If this rule is applied to the distance matrix in Figure 1, then the candidate TC_2 would be selected. Jiang et al. [10] also examined MaxMax and MaxAvg variants. In the presented enhanced version of ARP technique, the function f_2 is replaced with an MCDM method (see Section 3.1).

	Prioritized TC_1	Prioritized TC_2	Prioritized TC_3
Candidate TC_1	0.2	0.5	0.3
Candidate TC_2	0.8	0.4	0.1
Candidate TC_3	0.5	0.6	0.2

Figure 1
Distance matrix

In the last steps, the selected test case is removed from *UTS* and added into the prioritized test sequence *PTS*. (Part 1 – lines 9 and 10). This process is repeated until the prioritization process is finished (all the test cases are sorted).

2.2 Path Complexity

Kaur et al. [16] originally designed Path Complexity (PC) TCP technique for systems modeled as UML activity diagrams. Each test case is represented as a path through a Control Flow Graph (CFG), which is converted from a UML nested activity diagram. For each path *P* from a generated set, several properties are calculated:

- N_p - the number of nodes traversed by *P*
- W_p - weight of test path *P*
- P_p - number of predicate nodes traversed by *P*
- C_p - number of logical conditions traversed by *P*
- complexity *C* of *P* by using the formula $C = N_p + W_p + P_p + C_p$

Where the weight of the path is based on Information Flow metric (*IF*). The *IF* metric was designed by Sharma et al. [17] and originally applied to the components of system design. In our case, it is used on each node *N* in the CFG model, and it is calculated as:

$$IF(N) = FANIN(N) \times FANOUT(N) \tag{1}$$

where, *N* is a node from CFG, *FANIN*(*N*) is a number of incoming flows to *N*, and *FANOUT*(*N*) is a number of outgoing flows from N. The weight of the path is a sum of IF from all nodes in the path.

$$W_p = \sum_{i=1}^{n} w_i \forall P \in T_P \quad (2)$$

Where W_p is weight of the path *P*, w_i is weight of i^{th} node (*IF(N)*), *n* is a count of nodes in the current path *P*, and T_p is set of generated paths. When the complexity for all paths is calculated, the test cases (paths) are executed in order of complexity from highest to lowest.

2.3 Additional Coverage

The additional coverage technique is based on greedy reasoning applied to TCP [4]. The technique progresses iteratively through a test suite. In each step, a test case that yields the highest coverage of not yet covered statements, is selected.

2.4 Related Work

Code-based and regression testing are the most investigated TCP approaches [5, 6, 18, 19]. In the MBT area, proposed TCP techniques are frequently connected with UML (Activity) diagrams models [14, 20-22]. These techniques implement a wide range of strategies. Some of the strategies are modified variants from code-based context; they can be relatively straightforward as Path Complexity, or more advanced ones that include historical data and data mining techniques.

The proposed technique extends ARP technique, which is a general black-box technique. In this context, an interesting article was presented by Hemmati et al. [23]. The article compares three different black-box TCP (code-based) approaches: topic coverage, text diversity, and risk-driven heuristic. In the topic coverage TCP, topics are extracted from a textual description of test cases and their expected results by a text mining algorithm. In the subsequent step, the technique tries to prioritize test cases in a way to maximize coverage of those topics. The text diversity approach prioritizes test cases using a string distance between two text representations of the test cases. The risk-based approach uses information about detected faults in their previous executions for test case prioritization. If historical information is available, the results show that the risk-based approach is superior. However, none of the approaches significantly outperform the others, if history is not available.

Empirical study [24] performed by João Felipe Silva Ouriques et al. investigates the effect of the model structure and characteristics of test cases that fail on the fault detection capability of several TCP techniques. Study results show that the

characteristics of failed test cases affect the investigated techniques more significantly than the model layout. Due to the study being performed on synthetic data that may not exactly correspond to real systems, authors publish a replication study [8] on data from real industrial projects. In this study, multiple general TCP techniques were evaluated with similar results as in previous work. The results show that none of the compared techniques is the best, concerning fault detection ability. Besides, the study investigates a dependency between fault detection performance of particular techniques and different properties of test cases. Specifically, the effect of varying sizes of the test cases that fail is examined. Authors conclude that the examined techniques have different sensitivity to this test case feature. The presented article builds on these studies and presents a new enhanced version of ARP technique, which improves the fault detection performance of the standard version and preserves limited effect of this negative feature.

3 Novel Enhanced ARP Technique

The main objective of the novel enhanced ARP technique is to improve fault detection performance over the original ARP method. The ARP demonstrates consistent results in different scenarios thanks to its semi-random nature. The study results [8] show there are techniques with better fault detection performance, but their performance is also more affected by various test suite properties. One of those properties is an inconsistent fault detection performance between cases where failed test cases are shorter or longer than the average test case size in a test suite (see Section 4.1). The development of the proposed TCP technique was focused on fault detection performance enhancement while keeping relatively low sensitivity on the size of the test cases that fail.

The proposed modification replaces *Test set distance* function (function f_2 Algorithm 1 Part 3 – Line 8) with Weighted Product Model (MCDM) method [9]. The method compares candidates (test cases) among themselves, using distance, path complexity and additional coverage criteria. The novel enhanced TCP technique based on the MCDM method was named Multi-Criteria Adaptive Random Prioritization technique (MC-ARP). The advantage of MC-ARP is that the criteria can be extended/exchanged in a situation when a new source of information becomes available (e.g., fault history during testing of an updated system version).

The introduced MC-ARP technique partially decreases the importance of test case distances and increases the chance of selection for test cases which cover more not yet covered statements, or test cases with higher path complexity. The method can be tuned by weights, which determine the strength of these properties and thus

also results of the prioritization. The novel MC-ARP technique is described in the following subsection.

3.1 Multi-Criteria ARP Technique

The proposed application of Weighted Product Model method has *m* alternatives (not-yet prioritized test cases) and *n* decision criteria. The method compares alternatives among themselves, using these decision criteria, and determines the one that is better than others. The decision criteria are benefit criteria (higher values are better), and they are divided into three main groups:

- Distance criteria – performance value of a distance criterion corresponds to distance between a particular candidate and already sorted test cases (distance matrix – see Section 2.1)
- Path Complexity – path complexity value of the candidate (see Section 2.2)
- Additional coverage – represents the count of newly covered statements, if the candidate is selected (see Section 2.3)

Variable *w*, which determines the relative weight of importance of the criteria, is assigned to each of these criteria groups. Due to the fact that the count of prioritized test cases changes during the TCP process (each time when a new sorted test case is added), the weight of the individual criterion can be calculated as an equal share of initial weight for distance criteria w_{DC}.

$$w_i = \frac{w_{DC}}{n_{DC}} \tag{3}$$

Where w_i is weight for a specific distance criterion (prioritized test case), and n_{DC} is the number of prioritized test cases. Weights for path complexity w_{PC} and additional cover w_{AC} do not change during algorithm iterations. Thus, the ratio between all weights is always the same; only the values of w_i are iteratively changed.

The performance value of candidate test case TC_i during evaluation by criterion *j* is denoted as pv_{ij}. The following function $P(TC_K/TC_L)$ compares two candidates TC_K and TC_L. In case the result is higher than value 1, the first candidate is superior to the second. The newly selected candidate should be better than or at least equal to all other candidates.

$$P(TC_K/TC_L) = \prod_{j=1}^{n} \left(\frac{pv_{Kj}}{pv_{Lj}} \right)^{w_j}, \text{ for } K, L = 1, 2, 3 \ldots m; \text{ and } K \neq L \tag{4}$$

The example shows the method on model data. Let us say, there are three candidates and three already prioritized test cases, weights are set to w_{DC} = 0.5, w_{PC} = 0.2 and w_{AC} = 0.3, and the performance values of candidates are depicted in Figure 2, where columns represent particular criteria, the first row shows weights for a specific criterion, and other rows are connected to candidate test cases (alternatives). In our case, distance criteria performance values match to the distance matrix *d* from the original ARP example. However, other criteria are now considered in the test case selection.

The comparison of candidate TC_1 and TC_2 would be as follows:

$$P(TC_1/TC_2) = (0.2/0.8)^{\frac{1}{6}} * (0.5/0.4)^{\frac{1}{6}} * (0.3/0.1)^{\frac{1}{6}} * (10/15)^{0.2} * (3/1)^{0.3} \cong 1.33$$

Similarly, $P(TC_1/TC_3) \cong 0.79$ and $P(TC_2/TC_3) \cong 0.62$. Therefore, the selected candidate is TC_3 (instead of TC_2 from the ARP example), since it is better than all of the other candidates.

	Distance criteria				
	Prioritized TC_1	Prioritized TC_2	Prioritized TC_3	Path Complexity	Additional Cover
Weights	1/6	1/6	1/6	0.2	0.3
Candidate TC_1	0.2	0.5	0.3	10	3
Candidate TC_2	0.8	0.4	0.1	15	1
Candidate TC_3	0.5	0.6	0.2	12	4

Figure 2
MC-ARP Example

4 Experiments

In this section, it is verified whether the proposed MC-ARP technique achieves better fault detection results than the original ARP. The second goal of the experiments is to investigate the effect of the size of the test cases that fail on fault detection performance of the proposed technique. The experimental evaluation on a broader sample of test suites is important for the assessment of the newly proposed technique, because TCP techniques may have an outstanding performance for one test suite and weak results for another.

The section is divided into two main parts. The first part describes the experimental setup and other necessary aspects for the evaluation of the experiments. In the second part, the performance comparison of MC-ARP with the original ARP technique and investigation of sensitivity to the size of the test cases that fail, is presented.

4.1 Dataset

Dataset used in this section for evaluation of the proposed technique is obtained from [25]. The dataset includes 17 test suites and information about faults from six industrial systems (for overall characteristics see Table 1). Each system is covered by two to four test suites, and each test suite has 4 to 24 test cases (for more information see [8]). The projects included in the dataset are from different areas, e.g., a cashdesk system that interacts with payment terminals, or a system to manage lending of equipment/software and maintenance logs. The tested systems were modeled in a high abstraction level as control-flow Labelled Transition System models. The models represent use scenarios, which can include multiple types of control flows (standard scenario, alternative user's behavior, and exception flow that covers systems errors). Test cases were generated as paths through these models by traverses of a Depth-First Search algorithm.

Table 1
Systems overall characteristics [8]

System	Language	Size (LOC)
S1	Java	3000
S2	C	3055
S3	Java	13001
S4	Groovy grails	3693
S5	Groovy java	20713
S6	Groovy JavaScript grails	13244

Sensitivity to the size of the test case that has found a fault is defined by a relation between the sizes of test cases that fail and the rest of the test suite. The relation divides test cases into two groups. The first group contains short test cases that

execute fewer steps than the average number (test cases that commonly do not traverse loops). In contrast, the second group consists of long test cases, which perform more system steps than the average. In order to evaluate this sensitivity, it is necessary to know the failed test cases in advance. Then the test suites can be divided into the following groups:

- ShortTC – test suites where every test case that fails is shorter than the average size of test case in the test suite.
- LongTC – test suites where every test case that fails is longer than the average size of test case in the test suite.
- ConstantSizeTC – test suites where all test cases have the same size.
- MixedTC – test suites which do not fit to above mentioned groups.

The terminology is taken from the original study [8], and the distribution of these groups in the dataset is shown in Figure 3.

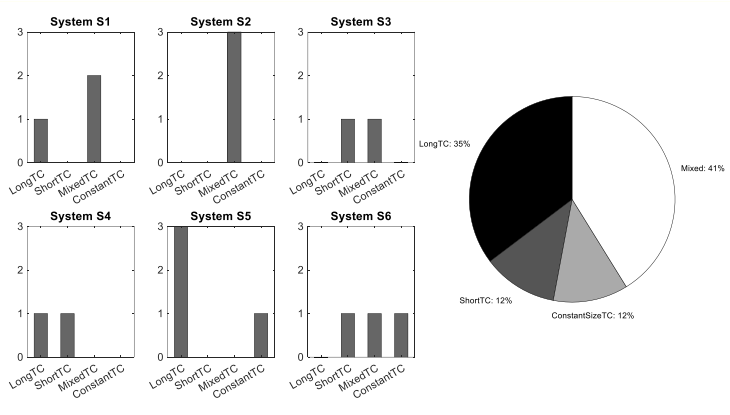

Figure 3
Test Suites Distribution

Random nature of ARP technique can affect TCP results. Therefore, the examined algorithms were executed 1000 times for each test suite (according to the suggestion in [26]); the experimental setup is shown in Figure 4.

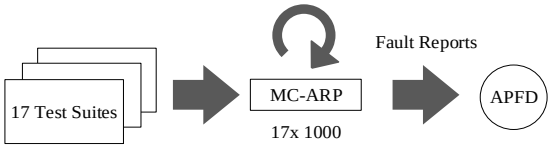

Figure 4
Setup overview

4.2 Effectiveness Metrics

To evaluate the performance of the new enhanced ARP technique and to compare results with the original methods, the following metrics are used.

Average Percentage of Faults Detection (APFD) developed by Elbaum et al. in [27] is used for evaluation of fault detection performance. The metric measures the rate of fault detection per percentage of test suite execution and is calculated as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + ... + TF_m}{mn} + \frac{1}{2n} \tag{5}$$

where n is the number of test cases, and m is the number of faults which the test suite can reveal. The TF_i is the position of the first test case that reveals the i–th fault. The APFD is a percentage (values 0 – 100), and higher values indicate better (faster) fault detection.

Non-parametric statistical test, Kruskal-Wallis test [28], is applied to compare two distributions of the APFD results. The test determines whether the difference between the results is statistically significant using a 95% confidence level (i.e. p-value < 0.05). The test resolves whether the differences are not random, but for performance comparison Vargha and Delaney's $\hat{A}_{12}$ statistic [29] is used.

Vargha and Delaney's $\hat{A}_{12}$ statistic performs a pairwise comparison of results expressed by the APFD metric from two techniques A and B. It is a nonparametric effect size measure, which is popular in the software engineering area, where randomized algorithms are involved [30]. The $\hat{A}_{12}$ metric measures the probability that running A produces a higher APFD than running B. The $\hat{A}_{12}$ can be calculated:

$$\hat{A}_{12} = \frac{\frac{R_1}{m} - \frac{m+1}{2}}{n} \tag{6}$$

Where R_1 is the rank sum of the APDF results from the first compared technique (the ranking is done through the results of both techniques). The m is the number of results from the first technique, and n is the number of results from the second technique. If the two techniques are equivalent, then $\hat{A}_{12} = 0.5$. In another case, one technique produces better results.

4.3 Results

The performance evaluation of the presented technique was done by comparison of fault detection capabilities of the original and enhanced technique. For evaluation, several variants of ARP technique with the following *Test case*

distance functions were chosen: Jaccard (**ARP_{Jac}**), Manhattan (**ARP_{Man}**), and Similarity function (**ARP_{Sim}**). The original technique with Jaccard and Manhattan uses MaxMin *Test set distance* function, and Similarity function uses MaxMin and MaxMax *Test set distance* functions (marked **ARP_{Sim1}** and **ARP_{Sim2}**). The proposed Multi-criteria ARP technique is marked **MC-ARP_{XXX}**, where XXX distinguishes a specific variant with appropriate *Test case distance* function. The criteria weights for MC-ARP were experimentally set to: $w_{DC} = 0.5$, $w_{PC} = 0.2$ and $w_{AC} = 0.3$.

The overall fault detection results of MC-ARP and original variants are presented in Figure 5, and results for individual systems are shown in Figure 6. The pairwise Â12 results comparison can be found in Table 2.

Table 2
Effect sizes of pairwise comparisons of MC-ARP and the original technique

	ARP_{Jac}	ARP_{Man}	ARP_{Sim1}	ARP_{Sim2}
System S1	0.69	0.68	0.58	0.55
System S2	0.70	0.64	0.68	0.68
System S3	0.51	0.5	0.51	0.58
System S4	0.59	0.58	0.70	0.71
System S5	0.58	0.56	0.62	0.62
System S6	0.62	0.64	0.53	0.58
Overall	**0.59**	**0.58**	**0.59**	**0.61**

The overall comparison in Figure 5 shows that, in all cases, MC-ARP variants have higher median value (better results), and boxplots are also more compact (i.e., they have shorter interquartile ranges and more consistent results). Presented results from Table 2 show that MC-ARP improves the overall performance of ARP technique, and the improved fault detection performance is similar for each distance function variant.

Results for individual systems indicate that the more significant performance improvement is evident in systems S1, S2, and S4. These systems mostly contain MixedTC and LongTC test suites (see Figure 3). This improvement is due to Path Complexity and Additional Coverage criteria, which prefer more complex test cases with a higher amount of non-covered statements. In other cases, MC-ARP technique still has the same or better results than the original ARP variants i.e., boxplots are more compact or median values are similar or higher.

For overall results, Kruskal-Wallis test produces the highest p-value < 0.001. Therefore, the techniques present different performances than their original versions. For individual systems, the results are statistically similar (p-value > 0.05) only in the case of the system S3 for techniques **ARP_{Jac}**, **ARP_{Man}**, and **ARP_{Sim1}**.

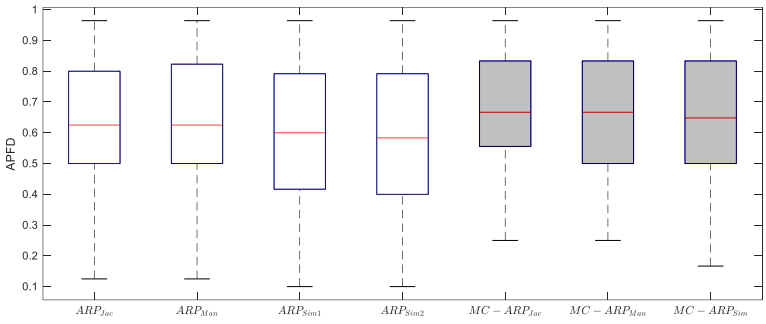

Figure 5
Comparison of overall results

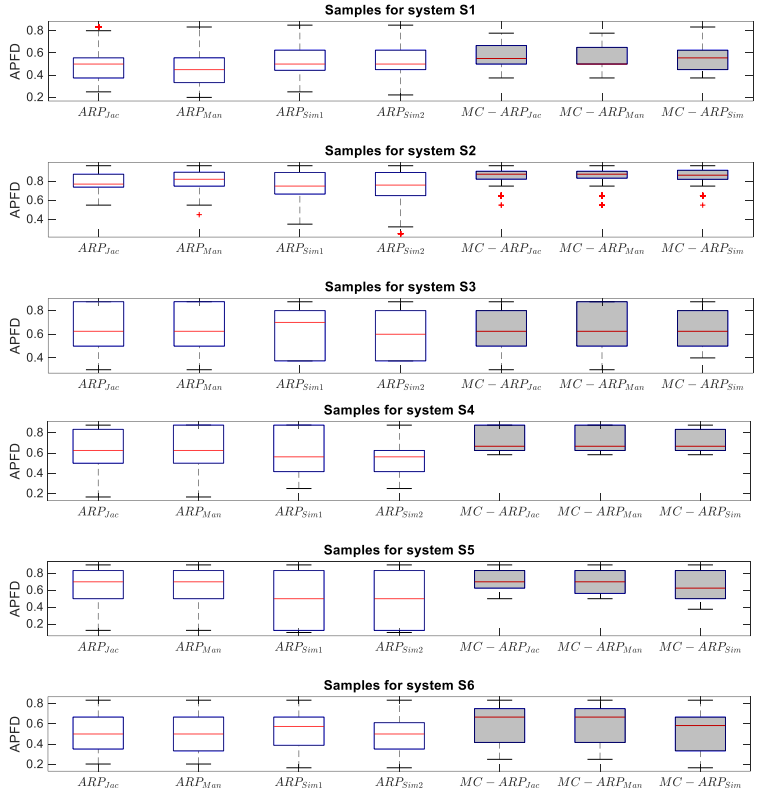

Figure 6
Performance of techniques for individual SUT

The effect of the size of the test cases that fail is investigated using ShortTC and LongTC suite samples. These samples contain test suites where failed test cases are shorter or longer than the average test cases size in that test suite.

Figure 7 presents the results for evaluated techniques. Detailed pairwise comparison is in Table 3 for ShortTC, respectively LongTC. The results show that the MC-ARP has better fault detection performance than the original ARP technique (mainly due to path complexity guidance) for LongTC samples. The MC-ARP variants have the same or higher median values and more compact boxplots. On the other hand, for ShortTC samples, Table 3 presents that MC-ARP has only slightly better performance than the original technique for Jaccard and Manhattan distance functions. The MC-ARP variant with Jaccard function has a higher median, but the boxplot is more spread out. The Manhattan variants have similar boxplots; however MC-ARP achieve higher median value. In case of Similarity function, the results are worse than the original technique, which is mainly noticeable on the boxplot for **ARP_{Sim1}**.

In the comparison of MC-ARP and the original variants, Kruskal-Wallis test results reach maximal p-value < 0.001. Hence, the techniques also produce different performance results.

Table 3
Effect sizes of pairwise comparisons of MC-ARP and the original ARP for ShortTC and LongTC

ShortTC	ARP_{Jac}	ARP_{Man}	ARP_{Sim1}	ARP_{Sim2}
$MC\text{-}ARP_{XXX}$	0.54	0.54	0.33	0.42
LongTC	ARP_{Jac}	ARP_{Man}	ARP_{Sim1}	ARP_{Sim2}
$MC\text{-}ARP_{XXX}$	0.60	0.57	0.60	0.60

Pairwise Â12 between ShortTC and LongTC for a technique represents sensitivity to the size of the test cases that fail. The technique insensitive to the size of the test cases that fail should reach a value 0.5 when performance for both groups is the same. The results for ARP and MC-ARP techniques are presented in Table 4. At this point, the proposed technique achieved a minor decrease compared to the original ARP values. This increase of sensitivity is caused by an unequal improvement of results between ShortTC and LongTC samples. However, the sensitivity of MC-ARP still achieves decent values in comparison to other TCP techniques (for more information about these techniques see [8]).

Records for ARP technique variants are duplicated in Table 4. Values based on data from our ARP implementation have the blue italic font. The results of the original study are listed in black font. These values have been added, because they are slightly dissimilar to the original, probably due to a different implementation of the ARP algorithm.

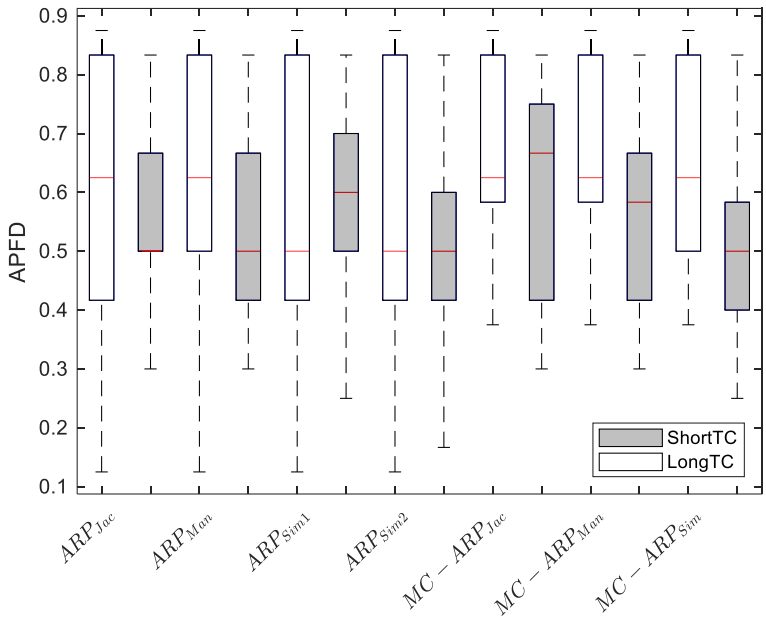

Figure 7
Boxplots of ShortTC and LongTC samples

Table 4
Effect sizes of the comparisons between ShortTC and LongTC [8]

Technique	Â12	Technique	Â12
Ran	0.4443	PC	0
ARP_{Jac}	0.3945	Stoop	0.5833
ARP_{Man}	0.374	SD_h	0.0833
ARP_{Sim1}	0.4725	SD_e	0.1666
ARP_{Sim2}	0.3892	SD_m	0
ARP_{Jac}	*0.44*	ST	1
ARP_{Man}	*0.4*	SA	0.1609
ARP_{Sim1}	*0.51*	**MC-ARP_{Jac}**	**0.35**
ARP_{Sim2}	*0.4*	**MC-ARP_{Man}**	**0.34**
FW	1	**MC-ARP_{Sim}**	**0.22**

Conclusions

This work presents a new Adaptive Random Prioritization technique (referred to as MC-ARP); the technique replaces the original *Test set distance* function, with a Multi-Criteria Decision-Making method. This enhancement incorporates additional criteria (other than the test case distances) into a decision which test case should be selected from the candidate set. The key idea of ARP technique

performance improvement is to add guidance based on the path complexity and additional coverage techniques that have better overall fault detection performance. The PC technique calculates a score based on a path through a control flow model for each test case and the test cases with a higher score, are preferred over the others. An additional cover technique calculates how many yet, uncovered statements, will be covered when a test case will be added to the set of already the prioritized ones. The test case with the highest number of newly covered statements is preferred.

The mentioned features and distances between a candidate and already prioritized test cases are used as decision criteria in Weighted Product Model method. The method performs a pairwise comparison of all candidates. The proposed change partially limits the random nature of ARP and prefers to select more complex test cases or test cases with more uncovered statements over others.

The novel *Test set distance* function helps to improve the fault detection performance of ARP technique. Improvement of fault detection performance across all tested systems has been noticed. Moreover, experiment results show that MC-ARP has the same or better performance across all systems than the original ARP.

For test suites where every test case that fails is shorter than the average size (i.e., the test cases with less complex paths), MC-ARP with Jaccard, or Manhattan distance functions achieve slightly better results. Thanks to the guidance, MC-ARP outperforms the original technique when every test case that fails is longer than the average size of test cases in the test suite. The resulting sensitivity to the size of the test cases that fail is moderately higher than the original technique, due to uneven performance improvements in the scenarios mentioned above. However, it still achieves good results compared to other techniques.

Regarding threats to validity, the evaluation of the proposed MC-ARP technique was performed on the dataset that contains data from six industrial projects. The systems were modeled as Labelled Transition, and the test suites were generated by Depth-First Search algorithm, which traverse the models and saves paths as test cases. Therefore, the results cannot be generalized for other kinds of systems or other test case generation algorithms. However, it can be assumed, that for similar systems and test case generation approaches, the technique will perform at similar performance level. The evaluation was done only on fault detection and the effect of the size of test cases that fail, however, other aspects as the number of test cases in the dataset or the proportion of test cases that fail may affect the fault detection performance. Moreover, some test suites in the dataset are relatively small, and they may not be entirely suitable for prioritization.

In future work, the technique herein will be applied in the area of automotive Hardware-in-the-Loop (HIL) Integration Testing [31]. In this domain, MBT approach is used for Integration Testing of Electronic Control Units (ECUs). The goal of this testing phase is to estimate if a cluster of ECUs operates in synergy

and functions distributed among multiple ECUs work as expected. There is a need for optimization of automatically generated test suites. Those test suites are generated from Timed Automata models using our testing tool called Taster [32, 33]. The size of a test suite depends on a Timed Automata model complexity and used a state-space traversal algorithm. In general, it can be enormous. The MC-ARP will be implemented as part of this software. The expectation is to attain test suite optimization, towards shorter test times, while maintaining a reasonable test coverage. Further experiments will be performed on a HIL testing platform, in cooperation with our Industrial Partner. This HIL testbed is dedicated to the testing of comfort and radar sensor-based driver-assistance systems.

Acknowledgement

The work was supported from ERDF/ESF project "Advanced Testing of Automotive Radars" (No. CZ.02.1.01/0.0/0.0/16_025/0007318). This support is gratefully acknowledged.

References

[1] Ammann, P., J. Offut: Introduction to software testing, 1 ed., Cambridge University Press, New York, NY, USA, 2008

[2] Utting, M., B. Legeard: Practical model-based testing: a tools approach. Elsevier, 2010

[3] Zander, J., I. Schieferdecker, and P. J. Mosterman: A taxonomy of model-based testing for embedded systems from multiple industry domains, Model-based testing for embedded systems, 2011, pp. 3-22

[4] Elbaum, S., A. G. Malishevsky, and G. Rothermel: Test case prioritization: A family of empirical studies. IEEE transactions on software engineering, 2002, 28(2): pp. 159-182

[5] Catal, C. and D. Mishra: Test case prioritization: a systematic mapping study. Software Quality Journal, 2013, 21(3): pp. 445-478

[6] Khatibsyarbini, M., et al.: Test case prioritization approaches in regression testing: A systematic literature review. 2018, 93: p. 74-93

[7] Stallbaum, H., A. Metzger, and K. Pohl: An automated technique for risk-based test case generation and prioritization. in Proceedings of the 3rd international workshop on Automation of software test. 2008

[8] Ouriques, J. F. S., E. G. Cartaxo, and P. D. Machado: Test case prioritization techniques for model-based testing: a replicated study. Software Quality Journal, 2017, pp. 1-32

[9] Triantaphyllou, E.: Multi-criteria decision making methods, in Multi-criteria decision making methods: A comparative study. 2000, Springer, pp. 5-21

[10] Jiang, B., et al.: Adaptive random test case prioritization. in Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering. 2009

[11] Chen, T. Y., H. Leung, and I. Mak: Adaptive random testing. in Annual Asian Computing Science Conference. 2004, Springer

[12] Jaccard, P.: Comparative study of the floral distribution in a portion of the Alps and the Jura Mountains (Étude comparative de la distribution florale dans une portion des Alpes et des Jura) 1901, 37: pp. 547-579

[13] Zhou, Z. Q.: Using coverage information to guide test case selection in adaptive random testing. in 2010 34th Annual IEEE Computer Software and Applications Conference Workshops. 2010

[14] Zhou, Z. Q., A. Sinaga, and W. Susilo: On the fault-detection capabilities of adaptive random test case prioritization: Case studies with large test suites. in System Science (HICSS), 2012 45th Hawaii International Conference on. 2012

[15] Coutinho, A. E. V. B., E. G. Cartaxo, and P. D. de Lima Machado: Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing. Software Quality Journal, 2016, 24(2): pp. 407-445

[16] Kaur, P., P. Bansal, and R. Sibal: Prioritization of test scenarios derived from UML activity diagram using path complexity. in Proceedings of the CUBE International Information Technology Conference. 2012

[17] Sharma, C., S. Sabharwal, and R. Sibal: Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams. arXiv:1410.4838, 2014

[18] Kumar, A. and K. J. C. Singh: A Literature Survey on test case prioritization. 2014, 3(5): p. 793

[19] Korel, B., L. H. Tahat, and M. Harman: Test prioritization using system models. in Software Maintenance, 2005. ICSM'05. Proceedings of the 21st IEEE International Conference on. 2005

[20] Mahali, P., D. P. J. I. J. o. S. A. E. Mohapatra, and Management: Model based test case prioritization using UML behavioural diagrams and association rule mining. 2018, 9(5): pp. 1063-1079

[21] Kundu, D., et al.: System testing for object-oriented systems with test case prioritization. Software Testing, Verification Reliability, 2009, 19(4): p. 297-333

[22] Sapna, P. and H. Mohanty: Prioritization of scenarios based on uml activity diagrams. in Computational Intelligence, Communication Systems and Networks, 2009, CICSYN'09, First International Conference on. 2009

[23] Hemmati, H., Z. Fang, and M. V. Mantyla: Prioritizing manual test cases in traditional and rapid release environments. in Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. 2015

[24] Ouriques, J. F. S., et al.: Revealing influence of model structure and test case profile on the prioritization of test cases in the context of model-based testing. 2015. 3(1): p. 1

[25] Ouriques, J. F. S.: Replication of Failure Characteristics Experiment; Available from:https://sites.google.com/site/joaofso/research/experiments/replication-of-failure-characteristics-experiment

[26] Arcuri, A. and L. Briand: A practical guide for using statistical tests to assess randomized algorithms in software engineering. in Software Engineering (ICSE), 2011 33rd International Conference on. 2011

[27] Elbaum, S., A. G. Malishevsky, and G. Rothermel: Prioritizing test cases for regression testing. Vol. 25, 2000

[28] Sheskin, D. J.: Handbook of parametric and nonparametric statistical procedures. crc Press. 2003

[29] Vargha, A. and H. D. Delaney: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational Behavioral Statistics

[30] Poulding, S. and J. A. Clark: Efficient software verification: Statistical testing using automated search. IEEE Transactions on Software Engineering, 2010. 36(6): pp. 763-777

[31] Sobotka, J. and J. Novak: Automation of automotive integration testing process. in 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS) 2013

[32] Krejci, L. and J. Novak: Model-based testing of automotive distributed systems with automated prioritization. in Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2017 9th IEEE International Conference on. 2017

[33] Sobotka, J. and L. Krejci: Testing of Automotive Systems-Complex vs. Simple Environment Models. in 2018 16th Biennial Baltic Electronics Conference (BEC). 2018