Towards Earlier Fault Detection by Value-Driven Prioritization of Test Cases Using Fuzzy TOPSIS

Sahar Tahvili[1,2], Wasif Afzal[2], Mehrdad Saadatmand[1,2], Markus Bohlin[1], Daniel Sundmark[2] and Stig Larsson[1]

[1] SICS Swedish ICT, Västerås, Sweden
{sahart, mehrdad, markus.bohlin, stig.larsson}@sics.se
[2] Mälardalen University, Västerås, Sweden
{wasif.afzal, daniel.sundmark}@mdh.se

Abstract. Software testing in industrial projects typically requires large test suites. Executing them is commonly expensive in terms of effort and wall-clock time. Indiscriminately executing all available test cases leads to sub-optimal exploitation of testing resources. Selecting too few test cases for execution on the other hand might leave a large number of faults undiscovered. Limiting factors such as allocated budget and time constraints for testing further emphasizes the importance of test case prioritization in order to identify test cases that enable earlier detection of faults while respecting such constraints. This paper introduces a novel method prioritizing test cases to detect faults earlier. The method combines TOPSIS decision making with fuzzy principles. The method is based on multi-criteria like fault detection probability, execution time, or complexity. Applying the method in an industrial context for testing a train control management subsystem from Bombardier Transportation in Sweden shows its practical benefit.

Keywords: Software Testing, Fault Detection, Test Cases Prioritization, Optimization, Fuzzy Logic, MCDM, TOPSIS, Failure Rate

1 Introduction

Value-based software engineering [1] emphasizes the importance of integrating value considerations in software development. In software testing, prioritizing by value has immediate benefits, given that a lot of time and effort is spent on testing, rework and fixing of faults. In this paper, we use a value-based method of test case prioritization whereby we combine a multi-criteria decision making (MCDM) technique with fuzzy logic. The MCDM technique used is called TOPSIS (Technique for Order of Preference by Similarity to Ideal Solution) [2] that is combined with fuzzy principles and hence called as FTOPSIS [3]. To optimize for value, we have considered multiple factors but start with factors most important from a customer perspective: quality and time-to-market. In order to maximize quality, we want to execute test cases having higher fault detection probability while in order to reduce time-to-market, we need to minimize time in terms of reduced number of test executions. Still, we want to satisfy additional criteria such as cost and requirements coverage but after having optimized for the most important ones. The advantages of using TOPSIS include its ease of use, simplicity and ability to keep constant steps regardless of the problem size [4]. Using fuzzy principles with TOPSIS suits the problem of test case prioritization which is characterized by interplay of complex factors and difficulties in gathering precise data. Considering the time and cost required to execute each test case from a test suite, the ideal situation is to detect the same number of faults (as detected by the whole test suite) by executing as few test cases as possible and under minimum amount of time. To achieve this goal, prioritization of test cases based on their fault detection probability and execution time is required. However, other test case properties such as the number of requirements that they cover may also serve as additional prioritization criteria. Properties such as these are important for testers and thus need to be considered as well. Therefore, in our optimization problem to find the closest solution (i.e., a set of test cases) to the aforementioned ideal situation, such other criteria play a role and can affect the result. It is naturally beneficial to use a multi-criteria technique for solving such a prioritization problem. This paper, proposes such an approach for prioritizing a set of test cases for integration testing by using TOPSIS. Our approach enables to identify a set of test cases which are closest to the ideal situation, and therefore, contribute towards the ultimate goal of identifying faults earlier and under a shorter time. To facilitate the application of our approach in real world scenarios where precise specification of quantified values for different criteria may not always be possible, we apply and combine fuzzy concepts with TOPSIS to enable the specification of values using linguistic variables (i.e., high, low, etc.). Finally, to demonstrate and evaluate the applicability of our approach, we have applied it on a train control management subsystem from Bombardier Transportation AB, Sweden, hereby called as BT.

2 Background & Preliminaries

Today, there are several aspects of optimization in the testing process with different goals, such as increasing fault detection rate [5], decreasing the use of redundant test cases, or use resources more efficiently. By prioritizing test cases, we are able to propose an order for executing test cases at different levels of testing process. In this order, every single test case will be ranked with a value, which can either be a “sharp” value, or, in the context of fuzzy logic, a linguistic value (e.g., low, high, etc.) As there are more than one criteria which can affect the ranking of test cases, a sensible approach is to consider several of the aforementioned objectives at the same time as a multi-objective optimization problem (e.g. maximizing the rate of fault detection and minimizing cost). To solve this kind of problem, several multi criteria decision making (MCDM) analysis techniques have been developed. MCDM techniques formulate the problems as the interdependency between various criteria and alternatives [4]. TOPSIS, originally developed by Hwang and Yoon (see [2]), is one such MCDM technique which has been shown useful when combined with a fuzzy approach, and when there are some conflicting and no commensurable criteria in the initial problem.

2.1 Motivating Example

As mentioned earlier, for using TOPSIS we need to identify the criteria which have effect on the alternatives. Some standard criteria for software testing might be time, cost, requirement coverage, etc. In reality, there are some additional limitations during the testing process, for example due to budget, time, or resources constraints. By proposing an optimal order for execution, we are able to execute test cases according to such constraints. Figure 1 illustrate a MCDM problem where 5 different criteria (C_1 to C_5) have a direct effect on every single alternative (test cases), of which there are 10 in total. As we can see in Figure 1, there are limitations on budget and time in the problem. In fact, testers can only execute a limited number of test cases before the deadline or budget limit is reached. 'Goal achieved' in Figure 1 represents that testers detect the expected number of faults after executing 8 test cases in this particular order. The last two test cases (T_9 , T_{10}) are thus considered as redundant.

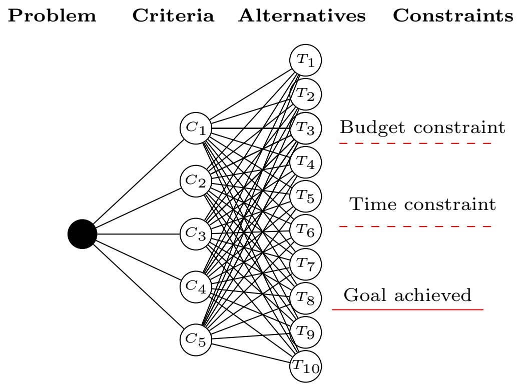

Fig. 1: Illustration of a MCDM problem with constraints.

3 Proposed Approach

The TOPSIS method is based on identification of two ideal solutions; the *positive ideal solution* (PIS) and the *negative ideal solution* (NIS). The PIS consists of the best value for each criteria (taken from the set of alternatives), and similarly, the NIS consists of the worst values for each criteria. The PIS, in the concept of prioritizing test cases, will then be an ideal test case which satisfies all the identified criteria properly, such as a fast (execution) and cheap (implementation cost) test case with very high probability of fault detection which tests a set of complex requirements. However, the negative ideal solution comprises slow and expensive test cases with very low probability of detecting the faults and just tests a simple single requirement. In TOPSIS, alternatives are then ranked according to their distance from the PIS (in increasing order of preference) and the NIS (in decreasing order of preference) simultaneously [6].

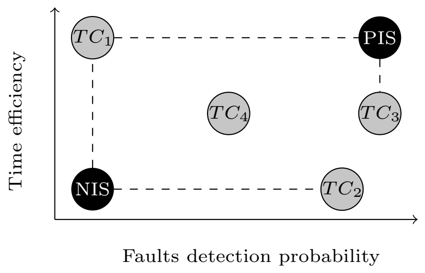

Fig. 2: Positive (PIS) and negative (NIS) ideal solutions in a bi-criteria prioritization problem for four test cases.

Figure 2 illustrates TOPSIS for two different criteria; time efficiency and fault detection probability. Measuring the effect of the criteria on the test cases is not an easy task as some criteria such as execution time, can be measured by a sharp value and some criteria like cost, can be written as a function of time or lines of code (or other measurable criteria). For estimating some other criteria (such as fault detection probability), software developers and testers can provide estimations using fuzzy linguistic variables by comparing test cases with each other. For this purpose, we later extend the initial problem to the fuzzy environment and use linguistic variables to capture the effect of weighted values with some uncertainty. It should be noted that this approach is not limited to any particular set of criteria and some other criteria can well be applied. After identifying a set of test cases that are closest to the positive ideal solution, we can calculate the fault failure rate (explained in Section 3.3) for this set which enables us then to compare them against other (sub-)set of test cases from the test suite or the whole test suite. Moreover, by using the fault failure rate, we can compare the efficiency of different set of test cases with respect to detecting faults and the number of test cases to execute to detect them. In short, our proposed approach consists of the following steps to enable earlier detection of faults through test case prioritization:

1. Identify the set of criteria for prioritization of test cases (besides fault detection probability and time efficiency).
2. Determine the criteria value for each test case using fuzzy linguistic variables. This can be done, for instance, by sending a questionnaire to testers.
3. Apply fuzzy TOPSIS in order to prioritize test cases and find a set of test cases closest to the positive ideal solution. During this process, consider the fault detection probability and time efficiency criteria as the most important among the set of criteria.

As an additional step and in order to compare and enable the evaluation of the identified set resulting from the application of FTOPSIS, fault failure rate can be calculated for the set of test cases. In the following sections, we describe the fuzzy TOPSIS method by using basic concept of fuzzy logic provided by Yang [7], Baets and Kerre [8] and Yun Shi [9].

3.1 Intuitionistic Fuzzy Sets (IFS)

Fuzzy set theory was proposed by Zadeh [10] for solving MCDM problems based on the inclusion degrees of Intuitionistic Fuzzy Sets (IFS) and a membership function. In this section, we define these formally.

Definition 1. *A fuzzy set is a pair (A, m_A) where A is a set and $m_A : A \to [0,1]$; for each $x \in A$, $m_A(x)$ is called the grade of membership of x in (A, m_A). Let $x \in A$, then x is fully included in the fuzzy set (A, μ_A) if $\mu_A(x) = 1$ and is fully excluded if $\mu_A(x) = 0$ where x is a fuzzy member if $0 < \mu_A(x) < 1$ (see[7]).*

The membership function can be illustrated by different shapes which helps interpreting the values appropriately. Triangular, bell-shaped, Gaussian and trapezoidal membership functions are the most common. In this paper, we use bell-shaped membership functions (see Figure 3) which is defined using the following formula:

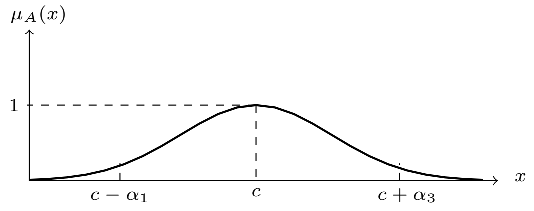

Fig. 3: Bell-shaped fuzzy membership function.

The membership of an element to a fuzzy set is a single value between 0 and 1 and can be obtained by:

$$\mu_{\tilde{A}}(x) = \frac{1}{1 + (x - \frac{c}{\alpha})^{2b}} \tag{1}$$

Since in reality, there is always a hesitation degree for calculating functional and non-functional degree, a generalization of fuzzy sets had been proposed as intuitionistic fuzzy sets (IFS) which incorporated the degree of hesitation called hesitation margin [11].

Definition 2. *An Intuitionistic Fuzzy Set (IFS) A on a universe U is defined as the following form: $A = \{(u, \mu_A(u), \nu_A(u)) \mid u \in U\}$, where the functions*

$u_A : U \to [0,1]$ and $v_A : U \to [0,1]$ define the degree of membership and the degree of non-membership of the element $u \in U$ in A, respectively, and for every $u \in U$ we have $0 \leq \mu_A(u) + \nu_A(u) \leq 1$, furthermore, $\pi_A(u) = 1 - \mu_A(u) - \nu_A(u)$ called the intuitionistic fuzzy set index or hesitation margin of u in A (see [11, 12]).

Thus, a fuzzy set can be written as:

$$\{(u, \mu_A(u), 1 - \mu_A(u)) \mid u \in U\} \tag{2}$$

IFS distributes fuzzy sets for every membership function μ and non-membership functions ν where $\nu = 1 - \mu$ (see [13]).

Definition 3. *A function $I : [0,1]^2 \rightharpoonup [0,1]$ is called a fuzzy implication if the following conditions are satisfied for all $x, y, z \in [0,1]$: if $x \leq y$ then $I(x,z) \geq I(y,z)$ and if $y \leq z$ then $I(x,y) \geq I(x,z)$ also the boundary conditions are: $I(0,0) = I(0,1) = I(1,1) = 1$ and $I(1,0) = 0$ (see [9]).*

A fuzzy implication can be calculated by using various formula such as Residuated implications (R-implications), Strong implications (S-implications) and Quantum Logic implications (QL-implications), in this work we use Residuated implications, proposed by Goguen [14]:

$$R_\pi(a,b) = \begin{cases} 1, & if\ a = 0, \\ \min\left(\frac{b}{a}, 1\right) & if\ a > 0 \end{cases} \tag{3}$$

Let U be a finite universe and R is an implication:

Definition 4. *Inclusion degree function of IFS denoted by I_{IFS}, if R satisfies the following conditions (see [9]):*

- *$\forall a, b \in [0,1]$ and $a \leq b \Rightarrow R(a,b) = 1$*
- *$R(a,b)$ is non-decreasing with respect to b and non-increasing with respect to a.*

thus:

$$I_{IFS}(A,B) = \frac{1}{|U|} \sum_{u \in U} [\lambda R_\pi(\mu_A(u), \mu_B(u)) + (1-\lambda) R_\pi(\nu_B(u), \nu_A(u))], \qquad \lambda \in [0,1] \tag{4}$$

where $|U|$ is the cardinality of U and calculated as (see [15]):

$$|U| = \sum_{u \in U} \frac{1 + \mu_A(u) - \nu_A(u)}{2} \tag{5}$$

3.2 Fuzzy TOPSIS

As previously mentioned, TOPSIS is based on positive (PIS) and negative (NIS) ideal solutions. The best alternative per time, is an alternative which has the farthest distance from NIS and the shortest distance from PIS.

Let $A = \{A_1, A_2, ..., A_n\}$ be a set of alternatives and $C = \{C_1, C_2, ..., C_m\}$ be a set of identified criteria. By using IFS, we are able to represent A and C as:

$$\begin{aligned}
A_1 &= \{(C_1, \mu_{1,1}, \nu_{1,1}), (C_2, \mu_{1,2}, \nu_{1,2}), ..., (C_m, \mu_{1,m}, \nu_{1,m})\} \\
A_2 &= \{(C_1, \mu_{2,1}, \nu_{2,1}), (C_2, \mu_{2,2}, \nu_{2,2}), ..., (C_m, \mu_{2,m}, \nu_{2,m})\} \\
&\vdots \\
A_n &= \{(C_1, \mu_{n,1}, \nu_{n,1}), (C_2, \mu_{n,2}, \nu_{n,2}), ..., (C_m, \mu_{n,m}, \nu_{n,m})\}
\end{aligned} \tag{6}$$

where $\mu_{i,j}$ indicates the degree by which the alternative A_i satisfies criterion C_j and $\nu_{i,j}$ indicates the degree by which the alternative A_i does not satisfy criterion C_j [13]. In FTOPSIS, the sum of $\mu_{i,j}$ and $\nu_{i,j}$ does not exceed 1.

Definition 5. *A fuzzy positive ideal solution is defined as:*

$$PIS_f = \{(C_1, \max\{\mu_{i,1}\}, \min\{\nu_{i,1}\}), \ldots (C_m, \max\{\mu_{i,m}\}, \min\{\nu_{i,m}\})\} \tag{7}$$

Definition 6. *A fuzzy negative ideal solution is defined as:*

$$NIS_f = \{(C_1, \min\{\mu_{i,1}\}, \max\{\nu_{i,1}\}), \ldots (C_m, \min\{\mu_{i,m}\}, \max\{\nu_{i,m}\})\} \tag{8}$$

The distance between the alternatives to PIS_f and NIS_f can be measured by inclusion degrees:

Definition 7. *The inclusion degree $D^+(A_i)$ of the positively ideal solution in alternative A_i is calculated by:*

$$D^+(A_i) = \max(I(PIS_f, A_i)), \tag{9}$$

and the inclusion degree $d^-(A_i)$ of the negatively ideal solution in alternative A_i is respectively measured as:

$$d^-(A_i) = \min(I(A_i, NIS_f)) \tag{10}$$

where I represents the inclusion degree function, calculated by Eq. (4).

Definition 8. *The ranking index of alternative A_i is defined as:*

$$P_i = \frac{D^+(A_i)}{d^-(A_i) + D^+(A_i)} \tag{11}$$

where $0 \leq P_i \leq 1$.

If there exists $i_0 \in \{1, 2, ..., n\}$ where $P_{i_0} = \max\{P_1, P_2, ..., P_n\}$, then A_{i_0} is the best alternative [13]. In fact, by selecting the maximum value of ranked index of alternatives (P_i) per mentioned criteria, we propose a set of alternatives which have satisfied the criteria properly. The ranking index of alternatives in the concept of prioritizing test cases, is a set of test cases (alternatives) which has maximum probability of detecting faults and also has a high time efficiency.

3.3 Fault Failure Rate

Definition 9. *Assume T test cases which are available for use, let F be the number of test cases that fail during the testing a system under test. Then the failure rate λ_{f_n} can be defined as the proportion of the failed test cases on the total number of executed test cases:*

$$\lambda_{f_n} = \frac{F}{T} \tag{12}$$

A high value for λ_{f_n} indicates that the faults are overall easy to detect, which implies that, the executed test cases have detected relatively more faults. However, a low value for λ_{f_n} shows that the faults in the system under test are harder to detect [16]. To calculate the failure rate, we have checked the initial version (the faulty version) of a set of test cases in integration testing level at BT.

4 Application OF FTOPSIS

As mentioned in Section 3, for prioritizing test cases using FTOPSIS, we need to identify a set of criteria having a direct effect on test cases. In consultations with our industrial partner (BT), the following criteria have been identified as the most effective ones:

– *Fault detection probability:* The probability of fault detection of test cases.
– *Time efficiency:* The sum of setup, implementation and execution time of each test case.
– *Cost:* The sum of cost incurred in implementation, hardware setup and configuration for each test case.
– *Requirement coverage:* The requirement(s) tested by each test case.

A set of 86 test cases, which test *Drive-Brake Control* and *Auxiliary Control* sub-level function groups in the train control management system at BT at the integration testing level, has been chosen as alternatives. To measure the effect of above-mentioned criteria on the test cases, a questionnaire was filled by a test expert at BT, using linguistic variables such as low, medium and high. To avoid lengthy calculations, we present a subset of test cases with ratings for different criteria in Table 1:

Test Case ID	Fault Detection	Time Efficiency	Cost	Requirement Coverage
HVAC-007	M	L	L	M
AirSupply-036	H	L	M	L
Drive-S-046	M	H	H	VH
Speed-S-IVV-005	M	M	VL	M
AirSupply-IVV-047	M	VH	M	M
SyTs-ExtDoors-S-IVV-011	H	M	L	M
⋮	⋮	⋮	⋮	⋮
Brake-041	VH	M	M	H

Table 1: The effect of criteria on test cases, with values very low (VL), low (L), medium (M), high (H) and very high (VH)

The specifications of the alternatives (test cases) and criteria for IFS based on Eq. (6) are calculated as follows (again we show it for a subset of test cases):

$$\begin{aligned}\text{HVAC-007} &=\{(C_1,0.5,0.5),(C_2,0.1,0.9),(C_3,0.1,0.9),(C_4,0.5,0.5)\}\\ \text{Airsupply-036} &=\{(C_1,0,9,0.1),(C_2,0.5,0.5),(C_3,0.5,0.5),(C_4,0.1,0.9)\}\\ &\vdots\\ \text{Brake-041} &=\{(C_1,0.9,0.1),(C_2,0.6,0.4),(C_3,0.5,0.5),(C_4,0.5,0.5)\}\end{aligned}$$

By using Eqs. (7) & (8), we calculate the positive and negative ideal solutions:

$$\begin{aligned}PIS_f &= \{(C_1,1,0.1),(C_2,0.8,0),(C_3,0.7,0.3),(C_4,0.9,0.1)\}\\ NIS_f &= \{(C_1,0.1,0.9),(C_2,0,1),(C_3,0.1,0.9),(C_4,0.1,0.8)\}\end{aligned}$$

The inclusion degrees of PIS and NIS are calculated by Eqs. (9) & (10) and the results have been summarized in Table 2.

	HVAC-007	Airsupply-036	Drive-S-046	...	Brake-041
$I(PIS_f, A_i)$	0.35	0.45	0.51		0.61
$I(A_i, NIS_f)$	0.78	0.41	0.48		0.45

Table 2: The inclusion degrees of PIS_f and NIS_f in A_i

Table 3 show the ranking indices (P_i) of test cases that are obtained by using Eq. (11). As we can see in Table 3, the test cases with ID number Brake-041

Test Case ID	HVAC-007	Airsupply-036	Drive-S-046	...	Brake-041
P_i	0.30	0.52	0.50		0.58

Table 3: The ranking index of test cases

has the maximum value ($P = 0.58$), which means that this test case should be executed first. As is evident from Table 1, fault detection probability for it is high and the mentioned test case has a higher time efficiency in comparison with HVAC-007, Airsupply-036 and Drive-S-046. Based on Table 3, we propose the following order of execution: {Brake-041 $\rightarrow$ Airsupply-036$\rightarrow$Drive-S-046 $\rightarrow \cdots$ $\rightarrow$ HVAC-007}. By calculating the ranking index for every single test case in Table 1, we are able to get a set of best candidates for execution that satisfy first fault detection probability and time efficiency and then rest of other criteria. We propose a subset of 48 test cases for execution (48 test cases of among total 86 test cases) which are are time efficient test cases and also have a high ability to detect the faults.

4.1 Industrial Evaluation

To evaluate our approach using an industrial case study, we have monitored the result of executing 86 test cases at the initial level of integration testing at BT.

In this level, 2 sub-level function groups, which are *Drive-Brake Control* and *Auxiliary Control*, have been tested by 86 test cases. Table 4 presents the results of monitoring the test effort.

Number of sub-level function groups	2
Executed test cases	86
Total passed test cases	69
Total failed test cases	7
Not set test cases	10
Fault failure Rate λ_{f_0}	0.092

Table 4: Integration test result at BT

As is shown in Table 4, to detect 7 faults in the initial level of integration testing at BT, 86 test cases have been executed, which implies that 69 test cases could not find any fault during the testing process. Total of 10 test cases have not been set up for execution, which indicates that there were errors in the test specifications and these 10 test cases could not even get started. According to BT's method for executing test cases, 'Not set test cases' (10 mentioned test cases in Table 4) will be tested in the next level of testing, which is system testing at BT. Table 4 also show the fault failure rate for the testing effort, as described in Section 3.3. The rate is obviously low since very few test cases failed.

Figure 4 shows the correlation observed for the time efficiency and fault detection probability for 86 test cases. Not all 86 test cases are distinctly visible in this figure as some of them are overlapping. The X-axis (horizontal axis) represents the probability of detecting fault per test case and the Y-axis (vertical axis) represents the time efficiency. Since probability is quantified as a number between 0 and 1 (where 0 indicates impossibility and 1 indicates certainty), X-axis is built up on a scale of 0 to 1. As explained earlier, the truth value of the fuzzy number lies in between 0 and 1, which have been used to show the scale of Y-axis in Figure 4.

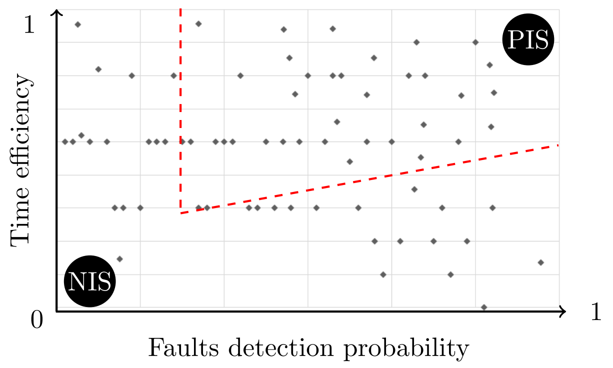

Fig. 4: Correlation for fault detection probability and time efficiency

We can see the positive ideal solution (PIS) and also the negative ideal solution (NIS), along with other test cases. As explained in Section 3, PIS represents a test case (or a set of test cases) that optimally satisfies the most important criteria. In Figure 4, 'fault detection probability' and 'time efficiency' have been illustrated as two critical criteria. In this situation, PIS is a test case (or a set of test cases) having highest probability of detecting faults and also being highly time efficient. As Figure 4 shows, some test cases have shorter distance to PIS and longer distance to NIS in comparison with other test cases and FTOPSIS identifies such test cases. The red dashed lines in Figure 4 represent the feasible set [3] of proposed test cases for execution. In other words, the test cases in the feasible set are satisfying both time efficiency and fault detection probability simultaneously. Other test cases which are not located in the feasible set have still a chance to detect some faults but they don't satisfy other criteria. In fact, by adding a new test cases in the proposed set for execution (48 test cases of among total 86 test cases) we are exiting from the feasible set, which means that, the newly joined test case (in the proposed set) is not able to satisfy at least on of the criteria. To evaluate the performance of our prioritized test cases, we need to measure the fault failure rate of prioritized test cases at regular intervals of time and compare it with current execution of test cases as shown in Table 4. For example, if we take 48 test cases to execute, using the prioritized set of test cases, we achieve a fault failure rate of $\lambda_{f_1} = 0.145$ (where 7 failed test cases are divided by 48 executed test cases). This is a significant improvement over the initial fault failure rate of $\lambda_{f_0} = 0.092$. It needs to be mentioned that some of the 'Not set test cases' (10 mentioned test cases in Table 4) are also in our proposed set for execution, but because these 10 test cases could not even get started, we are not able to use them for measuring fault failure rate. This result signifies that prioritization of test cases has immediate benefits in detecting faults early and hence improving software quality. We aspire to continue monitoring the fault failure rate at more regular time instances to better quantify the gains from prioritizing test cases.

5 Related Work

Prioritization of test cases in a test suite is about ordering their execution that increases test effectiveness, typically through maximizing early fault detection. This means that certain test cases are more valuable than others and prioritizing their execution will add value to the testing effort. Much of the literature on test case prioritization is in the context of regression testing, see e.g., the review paper by Yoo and Harman [17] and rely on code coverage based strategies. The motivation for using code coverage based strategies is that maximizing structural coverage will improve early maximization of fault detection. These strategies are of course not applicable for cases where such coverage information is either not readily available (e.g., in real-time embedded systems [18]) or is hard to trace (e.g., for testing at higher levels than unit). Other authors have used a requirements-based approach [19, 20] where test cases are prioritized by properties

[3]is the set of all possible points of an optimization problem that satisfy the problem's constraints

such as customer-assigned priority and implementation complexity. A criticism of such approaches is that such properties are subjective [17]. This has opened opportunities for using fuzzy approaches for prioritizing test cases since these approaches can better simulate experts' reasoning (see e.g., [21–26]). A fuzzy reasoning approach is also suited for a complex ranking problem as test case prioritization which is impacted by different criteria having uncertain, subjective and imprecise data. Fuzzy-based TOPSIS has previously been used in other domains (see e.g., [27]); in this paper we seek to evaluate its applicability in the context of test case prioritization. In this paper, we have used a fuzzy-based multi-criteria decision-making method known as FTOPSIS. Fuzzy-based TOPSIS has previously been used in other domains (see e.g., [27]); in this paper we seek to evaluate its applicability in the context of test case prioritization.

6 Conclusion and Future Work

Considering that the execution time for running various test cases are different, and also that each test case can have a different fault detection rate, it makes more sense to select the test cases for execution which have higher probability to detect faults and at the same time take shorter execution time. By prioritizing test cases based on such an order, it becomes possible to detect the faults in a system earlier. It should be noted that by prioritizing the test cases we are not able to detect more faults, yet the hidden faults in the system under test will be detected earlier. Using this method can be useful, for instance, in exploratory testing when time is severely limited to select and execute some random test cases to detect the hidden faults. In this paper, we introduced an approach based on the combination of fuzzy logic and TOPSIS multi-criteria decision making technique towards identifying a set of test cases that can detect faults with higher detection probability while at the same time result in a shorter overall execution time; hence earlier fault detection. We did this particularly by considering fault detection probability and time efficiency properties of each test case as two important criteria that are used in the decision making process. Through an example, we showed how the approach can be applied in practice. To enable the efficiency evaluation of the identified set of test cases, we used the concept of fault failure rate as an indicator to compare the fault detection capability of different sets of test cases. As a future work, we plan to perform more industrial case studies in order to evaluate more precisely how the identified set of test cases based on our approach and considering the estimated fault detection probability of test cases compares in practice against the test cases that have failed after executing the complete test suite. In other words, we are interested to perform more evaluations to see how our prediction in terms of suggesting a set of test cases matches the result of test suite after execution. From this perspective, one factor that can affect the outcome of our proposed approach is the estimation accuracy of fault detection probably and execution time of test cases which are provided by testers in a subjective manner, and thus are prone to human judgment errors. Investigation of methods and techniques that help in providing more accurate estimation of fault detection probability of test cases would be another interesting direction of this work. One observation that we had as part of this work was that the test cases that had higher fault detection probability, also targeted and covered more requirements as well. It would be interesting to investigate whether such correlation can also be observed in other systems and domains, and how existence of such correlation can help with better design of test cases to achieve a higher fault detection rate in a more efficient way. Another advantage of our proposed method is that the decision making part can be performed in an automatic and systematic way. However, the part that requires manual work and intervention is when values for various test criteria are to be specified. This can also be a limiting factor with respect to the scalability of our approach when the number of test cases for which various criteria values need to specified grows more and more. Identification of techniques and methods for more precise estimation of criteria values might help with the scalability issue but deserves a more thorough study and investigation in a separate work.

Acknowledgements

This work was supported by VINNOVA grant 2014-03397 through the IMPRINT project and the Swedish Knowledge Foundation (KKS) grant 20130085 through the TOCSYC project, and the ITS-EASY industrial research school.

References

1. Barry Boehm. Value-based software engineering. *SIGSOFT Software Engineering Notes*, 28(2):4–, 2003.
2. Ching-Lai Hwang and Kwangsun Yoon. Methods for multiple attribute decision making. In *Multiple attribute decision making*, pages 58–191. Springer, 1981.
3. Chen-Tung Chen. Extensions of the {TOPSIS} for group decision-making under fuzzy environment. *Fuzzy Sets and Systems*, 114(1):1 – 9, 2000.
4. Markand Velasquez and Patrick T. Hester. An analysis of multi-criteria decision making methods. *International Journal of Operations Research*, 10(2):56–66, 2013.
5. Gregg Rothermel, Roland H Untch, and Mary Jean Harrold. Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering*, 27(10):929–948, 2001.
6. Mehdi Amir-Aref, Nikbakhsh Javadian, and Mohammad Kazemi. A new fuzzy positive and negative ideal solution for fuzzy TOPSIS. *WSEAS Transactions on Circuits and Systems*, 11:92 – 103, 2012.
7. J. Yang and J. Watada. Fuzzy clustering analysis of data mining: Application to an accident mining system. *International Journal of Innovative Computing, Information and Control*, 8(8):5715–5724, 2012.
8. B. De Baets and E. Kerre. Fuzzy relational compositions. *Fuzzy Sets and Systems*, 60(1):109 – 120, 1993.
9. Baczyński Michał and Jayaram Balasubramaniam. *Fuzzy implications.* Springer-Verlag Berlin Heidelberg, 2008.

10. L.A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965.
11. P.A Ejegwa, S.O Akowe, P.M. Otene, and J.M Ikyule. An overview on intuitionistic fuzzy sets. *International Journal of Scientific and Technology Research*, 3(3):142 – 145, 2014.
12. Krassimir T. Atanassov. Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1):87 – 96, 1986.
13. Changrui Yu and Yan Luo. A fuzzy optimization method for multi-criteria decision-making problem based on the inclusion degrees of intuitionistic fuzzy sets. In De-Shuang Huang, II Wunsch, DonaldC., DanielS. Levine, and Kang-Hyun Jo, editors, *Advanced intelligent computing theories and applications. With aspects of artificial intelligence*, volume 5227 of *Lecture Notes in Computer Science*, pages 332–339. Springer Berlin Heidelberg, 2008.
14. Joseph A Goguen. The logic of inexact concepts. *Synthese*, 19(3):325–373, 1969.
15. Jiulun Fan, Weixin Xie, and Jihong Pei. Subsethood measure: new definitions. *Fuzzy Sets and Systems*, 106(2):201 – 209, 1999.
16. Debroy.V and Wong. W. Eric. On the estimation of adequate test set size using fault failure rates. *The Journal of Systems and Software*, page 587–602, 2011.
17. S. Yoo and M. Harman. Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability*, 22(2):67–120, 2012.
18. G. Wikstrand, R. Feldt, J.K. Gorantla, Wang Zhe, and C. White. Dynamic regression test selection based on a file cache: An industrial evaluation. In *Proceedings of the 2009 International Conference on Software Testing, Verification and Validation (ICST'09)*, 2009.
19. H. Srikanth, L. Williams, and J. Osborne. System test case prioritization of new and regression test cases. In *Proceedings of the 2005 International Symposium on Empirical Software Engineering (ESE'05)*, 2005.
20. R. Krishnamoorthi and S.A. Sahaaya Arul Mary. Factor oriented requirement coverage based system test case prioritization of new and regression test cases. *Information and Software Technology*, 51(4):799 – 808, 2009.
21. Sahar Tahvili, Mehrdad Saadatmand, and Markus Bohlin. Multi-criteria test case prioritization using fuzzy analytic hierarchy process. In *Proceedings of the 10th International Conference on Software Engineering Advances (ICSEA'15)*, 2015.
22. Ali M. Alakeel. Using fuzzy logic in test case prioritization for regression testing programs with assertions. *The Scientific World Journal*, 2014:1–9, 2014.
23. C. Malz, N. Jazdi, and P. Gohner. Prioritization of test cases using software agents and fuzzy logic. In *Proceedings of the 2012 IEEE 5th International Conference on Software Testing, Verification and Validation (ICST'12),*, 2012.
24. Zhiwei Xu, Kehan Gao, Taghi M. Khoshgoftaar, and Naeem Seliya. System regression test planning with a fuzzy expert system. *Information Sciences*, 259:532–543, 2014.
25. Charitha Hettiarachchi, Hyunsook Do, and Byoungju Choi. Risk-based test case prioritization using a fuzzy expert system. *Information and Software Technology*, 69:1–15, 2016.
26. A. Schwartz and Hyunsook Do. A fuzzy expert system for cost-effective regression testing strategies. In *Proceedings of the 2013 29th IEEE International Conference on Software Maintenance (ICSM'13)*, 2013.
27. Yeonjoo Kim, Eun-Sung Chung, Sang-Mook Jun, and Sang Ug Kim. Prioritizing the best sites for treated wastewater instream use in an urban watershed using fuzzy TOPSIS. *Resources, Conservation and Recycling*, 73:23 – 32, 2013.