Improved Metrics for Non-Classic Test Prioritization Problems

Ziyuan Wang\textsuperscript{1,2}  Lin Chen\textsuperscript{2}

\textsuperscript{1} School of Computer, Nanjing University of Posts and Telecommunications, Nanjing, 210003
\textsuperscript{2} State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, 210023

Email: wangziyuan@njupt.edu.cn

\textbf{Abstract}—The average percent of faults detected (APFD) and its variant versions are widely used to evaluate prioritized test suite’s efficiency. However, APFD is only available for classic test case prioritization, where all prioritized test suites under comparison contain the same number of test cases. If people overlook this phenomenon, they may obtain incorrect results in some non-classic scenarios, where prioritized test suites have different sizes. In addition, it can’t precisely illustrate the process of fault detection. Besides the APFD, most of its variants have similar problems. This paper points out these limitations in detail, by analyzing the physical explanation of APFD series metrics. To avoid limitations, a series of metrics including RAPFD, RAPFD\(C\), RAPFD\(W\) and RAPFD\(C\&W\) are proposed for different types of scenarios. All proposed metrics refer to both the speed of fault detection and the constraint of testing resource. There is an example in this paper showing that proposed metrics provide much more precise illustrations of fault detection process and fault detection efficiency of test suite.

\textbf{Keywords}—software testing, test case prioritization, fault detection efficiency, metric

I. INTRODUCTION

There is usually a contradiction in test case evolution. For the purpose of rapid version release, we usually need a speedup regression testing to save test resource (e.g. consumed time). However, for the purpose of higher-quality, we want to run test cases as many as possible. The contradiction between them tells people it is necessary to apply some test case optimization techniques to increase the effectiveness and efficiency of regression testing. As one of test case optimization techniques, the test case prioritization technique has been widely used to improve the efficiency of software testing.

The test case prioritization technique aims to schedule test cases in an order, to form a prioritized test suites. The classic test case prioritization problem is defined as follows. Giving an initial test suite \(T_{\text{init}}\), the test case prioritization technique aims to find a prioritized test suite \(\sigma \in PT\) such that:

\[
(\forall \sigma') (\sigma' \in PT) (\sigma \neq \sigma') [f(\sigma) > f(\sigma')]
\]

(1)

Where \(PT\) is the set of all permutations of \(T_{\text{init}}\) \((PT\) collects all possible prioritized test suites that contain all test cases in \(T_{\text{init}}\)). And \(f\) is an objective function from \(PT\) to real number as the award value of prioritized test suite \([1]\).

There could be many possible objective functions for test case prioritization problem. People usually restrict attention to the speed of detecting faults. Therefore, an objective function called \textit{average percent of faults detected (APFD)} is proposed as a metric to evaluate the speed of detecting faults \([1]\). It helps people to compare which prioritized test suite \(\sigma \in PT\) detects faults more rapidly. There are also some variants of APFD, including the \textit{normalized average percent of faults detected (NAPFD)} \([2]\), the \textit{cost-cognizant weighted average percent of faults detected (APFDW)} \([3]\), and etc. In this paper, we use the term \textit{APFD series metrics} to jointly call them.

APFD series metrics are designed for the classic test case prioritization problem that defined in Equation 1, which implies an assumption that all prioritized test suites in \(PT\) must contain all test cases in the initial test suite for each \(\sigma \in PT\), \(|\sigma| = |T_{\text{init}}|\). However, besides the classic problem, there may be some other non-classic test case prioritization problems, where above assumption does not hold:

(1) \textit{Time-aware test case prioritization} selects and prioritizes test cases under the time constraint \([6]\). Different selection algorithms may produce prioritized test suites with different sizes.

(2) \textit{Test goal prioritization} schedules test goals and generates test cases for important test goals earlier \([7]\). It may lead to different prioritized test suites because of the different orders of test goals.

(3) \textit{Test case re-generation prioritization strategy} incorporates test prioritization into test generation (e.g. in combinatorial testing \([5]\)). Different algorithms may generate prioritized test suites with different sizes.

(4) \textit{Test case reduction} \([8]\) and test case prioritization are often incorporated in test case optimization. Different test case reduction algorithms may output test suites with different sizes.

In these novel scenarios, heterogeneous candidate prioritized test suites may contain only partial test cases in initial test suite (e.g. time-aware test case prioritization and test case reduction), or sometimes be not concerned with any initial test suite at all (e.g. test case re-generation prioritization and test goal prioritization). APFD and existing variants can hardly work to evaluate and compare these candidate prioritized test suites, since the number of test cases that contained in each candidate prioritized test suites may be varying. And besides the limitation about test suites’ sizes, there is another limitation of existing \(APFD\) series metrics. That is they cannot precisely illustrate the process of fault detection in real world.

Therefore, we need some improved metrics for non-classic test case prioritization. In this paper, we propose an improved metric \textit{Relative-APFD} (RAPFD for short), which relates to a given testing resource constraint (determine how many test cases could be run), to replace \textit{APFD} and \textit{NAPFD}. And
furthermore, we discuss the test costs and fault severities, and propose the metric Relative-APFD\(_{CW}\) (RAPFD\(_{CW}\) for short) to replace APFD\(_C\). Examples show us that proposed metrics could provide much more precise illustrations of fault detection process and fault detection efficiency of test suite.

II. EXISTING APFD SERIES METRICS

Using notations that introduced in the ref. [6], we briefly introduce existing APFD series metrics in this section.

1) APFD: Let \(\sigma\) be a prioritized test suite under evaluation, \(\Phi\) the set of fault contained in the software, and \(TF(\phi, \sigma)\) the index of the first test case in \(\sigma\) that exposes fault \(\phi \in \Phi\). The APFD of \(\sigma\) is [1]:

\[
APFD(\sigma) = 1 - \frac{\sum_{\phi \in \Phi} TF(\phi, \sigma)}{|\sigma||\Phi|} + \frac{1}{2|\sigma|}
\] (2)

2) NAPFD: Sometimes, there may be non-detected fault that can’t be detected by any test cases in \(\sigma\). For each non-detected fault \(\phi \in \Phi\), Walcott et al. set \(TF(\phi, \sigma) = |\sigma| + 1\) as a penalty that may make APFD value to become negative [6]. To avoid the problem of negative award value, Cohen et al. set \(TF(\phi, \sigma) = 0\) and define an improved NAPFD [2]:

\[
NAPFD(\sigma) = p - \frac{\sum_{\phi \in \Phi} TF(\phi, \sigma)}{|\sigma||\Phi|} + \frac{p}{2|\sigma|}
\] (3)

Where \(p\) is the rate of faults detected by \(\sigma\):

\[
p = \frac{|\{\phi \in \Phi | TF(\phi, \sigma) \neq 0\}|}{|\Phi|}
\]

3) APFD\(_C\): Another improvement for APFD is to take the test costs and the fault severities into consideration. Let \(C_i\) be the cost of the \(i\)-th test case in \(\sigma\) \((i = 1, 2, ..., |\sigma|)\), \(S_\phi\) the severity of the fault \(\phi \in \Phi\). For the scenario where there is not any non-detected faults, the APFD\(_C\) is [3]:

\[
APFD_C(\sigma) = \frac{\sum_{\phi \in \Phi} (S_\phi \times (\sum_{i=1}^{|\sigma|} C_i - \frac{1}{2}CTF(\phi, \sigma)))}{\sum_{i=1}^{|\sigma|} C_i \times \sum_{\phi \in \Phi} S_\phi}
\] (4)

There is special case of APFD\(_C\), called APFD\(_{TA}\), for the scenario where fault severities are uniform [4].

4) NAPFD\(_C\): Similar to the APFD, the original version of APFD\(_C\) can not handle non-detected faults, since there is not any definition of \(TF(\phi, \sigma)\) for non-detected faults.

Here we propose the normalized cost-cognizant weighted average percent of faults detected (NAPFD\(_C\)), by defining \(TF(\phi, \sigma) = 0\) for each non-detected fault \(\phi \in \Phi\) and setting \(C_0 = 0\) for the dummy test case with index 0:

\[
NAPFD_C(\sigma) = p_c - \frac{\sum_{\phi \in \Phi} (S_\phi \times \sum_{i=1}^{|\sigma|} C_i)}{\sum_{i=1}^{|\sigma|} C_i \times \sum_{\phi \in \Phi} S_\phi} + \frac{(\sum_{i=1}^{|\sigma|} C_i - \frac{1}{2}CTF(\phi, \sigma))}{2 \times \sum_{i=1}^{|\sigma|} C_i \times \sum_{\phi \in \Phi} S_\phi}
\] (5)

Where \(p_c\) is the rate of total severities of faults detected by \(\sigma\):

\[
p_c = \frac{\sum_{\phi \in \Phi} S_\phi}{\sum_{\phi \in \Phi} S_\phi}
\]

It is evident that NAPFD\(_C\) will be equivalent to APFD\(_C\) in the scenarios where there is not any non-detected fault. And NAPFD\(_C\) will be equivalent to NAPFD in the scenarios where both test costs and fault severities are uniform.

III. PHYSICAL EXPLANATIONS OF EXISTING METRICS

For a test suite with \(|\sigma|\) test cases, we can use \(|\sigma| + 1\) discrete points to illustrate the relationship between the percent of faults detected (y-axis) and the percent of test cases run (x-axis). If we connect all these points by a curve, APFD and NAPFD show the area under the curve. E.g., we select 5 test cases from Table 1 to form a prioritized test suite \(\sigma_1 : T_3 - T_5 - T_2 - T_4 - T_1\). It detects 3 faults using 1 test case, 5 faults using 2 test cases, 7 faults using 4 test cases, and all 8 faults using all 5 test cases. So \(APFD(\sigma_1) = NAPFD(\sigma_1) = 0.6\) by drawing the curve that connects 6 discrete points \((0, 0), (\frac{1}{5}, \frac{3}{5}), (\frac{2}{5}, \frac{4}{5}), (\frac{3}{5}, \frac{7}{5}), (\frac{4}{5}, 1), (1, 1)\) (see Fig. 1).

**Table 1. Faults Detected by Test Cases**

<table>
<thead>
<tr>
<th>( \phi_1 )</th>
<th>( \phi_2 )</th>
<th>( \phi_3 )</th>
<th>( \phi_4 )</th>
<th>( \phi_5 )</th>
<th>( \phi_6 )</th>
<th>( \phi_7 )</th>
<th>( \phi_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T4</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td>x</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>T6</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td>x</td>
<td></td>
</tr>
</tbody>
</table>

**Fig. 1. APFD(\(\sigma_1\))**  **Fig. 2. Compare \(\sigma_1\) and \(\sigma_2\)**

The APFD\(_C\) and NAPFD\(_C\) have the similar physical explanations, if we replace the number of detected faults with the total severities of detected faults, and replace the number of test cases run with the costs that consumed by test cases.

Besides the above physical explanations, APFD series metrics can help people to control the risk of exceptional termination in testing process under the constraint of testing resource. As well known, during the software evolution, resource (including time) that distributed to software testing are often limited since the delay of develop, the deadline of release, and etc. So there is usually not enough time to run all test cases.

**Theorem 1.** Let \(C_i\) be the consumed time of the \(i\)-th test case \(T_i \in \sigma\) \((i = 1, 2, ..., |\sigma|)\), and \(S_\phi\) the severity of fault \(\phi \in \Phi\). Suppose the variable \(t \in [0, \sum_{i=1}^{|\sigma|} C_i]\) be the time
of the moment testing process terminates exceptionally. The \( NAPFD_C(\sigma) \) is the mathematical expectation (or expected value) of the percent of total severities of faults detected before the termination, if following two assumptions hold:

1) \( t \sim U[0, \sum_{i=1}^{\sigma_1} C_i] \): That is \( t \) follows the continuous uniform distribution with parameters \( (0, \sum_{i=1}^{\sigma_1} C_i) \).
2) During the running of the \( i \)-th test case \( T_i \in \sigma \), the total severities of its newly detected faults grow linearly with its consumed time.

Proof is omitted since the length constraint.

According to this theorem, testers can use \( NAPFD_C \) to select the prioritized test suite that make more contributions under the constraint of testing resource. As we claimed previously, \( APFD_C, NAPFD \) and \( APFD \) are all the special cases of \( NAPFD_C \), so they have the similar properties in scenarios where (1) \( p_c = 1 \), (2) costs / severities are uniform, (3) \( p = 1 \) and costs / severities are uniform, respectively.

IV. LIMITATIONS OF EXISTING METRICS

There are some limitations of \( APFD \) series metrics though they have practical explanations.

A. Test Suites’ Sizes or Total Costs

\( APFD \) series metrics are designed for the classic test case prioritization problem that defined in Equation 1, where all prioritized test suites under evaluation contain the same number of test cases. However, there may be some other types of scenarios in software evolution and testing evolution, including time-aware test case prioritization, test goal prioritization, test case re-generation prioritization, test case reduction, and etc, where people need to compare prioritized test suites in which the numbers of contained test cases are different. So there is a question: can \( APFD \) series metrics work in these scenarios?

There is a condition in Theorem 1 that \( t \sim U(0, \sum_{i=1}^{\sigma_1} C_i) \). Considering two prioritized test suites \( \sigma_1 \) and \( \sigma_2 \) where \( \sum_{i=1}^{\sigma_1} C_i \neq \sum_{i=1}^{\sigma_2} C_i \), there are \( t_{\sigma_1} \sim U(0, \sum_{i=1}^{\sigma_1} C_i) \) and \( t_{\sigma_2} \sim U(0, \sum_{i=1}^{\sigma_2} C_i) \) respectively. Though both \( t_{\sigma_1} \) and \( t_{\sigma_2} \) follow continuous uniform distribution, their parameters are different: the former follows the distribution with parameters \( (0, \sum_{i=1}^{\sigma_1} C_i) \) while the latter follows the distributions with parameters \( (0, \sum_{i=1}^{\sigma_2} C_i) \). It is unfair, and even meaningless, to compare mathematical expectation by using \( NAPFD_C \), when probability density functions are different. Therefore, \( APFD \) series metric are not suitable to be used to compare prioritized test suites with different sizes (for \( APFD \) and \( NAPFD \)) or different total costs (for \( APFD_C \) and \( NAPFD_C \)).

Here we can take the test cases and faults in Table 1 as examples to show some incorrect results when use \( APFD \) and \( NAPFD \). The incorrect results could be extended to illustrate the limitations of \( APFD_C \) and \( NAPFD_C \), since \( APFD \) and \( NAPFD \) are special cases of them respectively.

1) For situation that all faults are detected by prioritized test suites, construct two prioritized test suite \( \sigma_1 : T_3 - T_5 - T_2 - T_4 - T_1 \) and \( \sigma_2 : T_3 - T_5 - T_6 \). Note that both \( \sigma_1 \) and \( \sigma_2 \) detect all faults. Then we compare their \( APFD \) values (also see Fig. 2):

\[
APFD(\sigma_1) = APFD_C(\sigma_1) = \frac{3}{5} \quad APFD(\sigma_2) = APFD_C(\sigma_2) = \frac{1}{2}
\]

But, it is incorrect to say \( \sigma_1 \) is more efficient than \( \sigma_2 \). After run 1 (or 2) test case(s), both \( \sigma_1 \) and \( \sigma_2 \) detect 3 (or 5) faults; after run 3 test cases, \( \sigma_2 \) detects all 8 faults while \( \sigma_1 \) detects only 5; and finally \( \sigma_1 \) need 2 more test cases to detect all 8 faults. It is clear that \( \sigma_2 \) detects faults more rapidly than \( \sigma_1 \).

2) For situation that there are non-detected faults, construct two prioritized test suite \( \sigma_3 : T_3 - T_2 - T_5 \) and \( \sigma_4 : T_3 - T_2 \). Note that both \( \sigma_3 \) and \( \sigma_4 \) detect 5 faults. Then we get:

\[
NAPFD(\sigma_3) = \frac{17}{12} \quad NAPFD(\sigma_4) = \frac{11}{12}
\]

But, it is incorrect to say \( \sigma_3 \) is more efficient than \( \sigma_4 \). After run 1 test case, both \( \sigma_3 \) and \( \sigma_4 \) detect 3 faults; after run 2 test cases, \( \sigma_4 \) detects 5 faults while \( \sigma_3 \) detects 3 faults; and finally \( \sigma_3 \) need one more test case to detect 5 faults. It is clear that \( \sigma_4 \) detects faults more rapidly than \( \sigma_3 \).

This phenomenon is often overlooked. There may be some incorrect and confused experimental results in the applications of \( APFD \) series metrics in some previous papers. E.g. in ref. [6] and [2], authors used \( APFD \) and \( NAPFD \) respectively to compare prioritized test suites, in which the numbers of contained test cases are different, without any pretreatment.

B. Process of Fault Detection

\( APFD \) series metrics can’t precisely illustrate the process of fault detection in real world.

Note that the second condition of Theorem 1 is that the total severities of detected faults grow linearly with consumed time. In detail, during the running of one given test case, the number of newly detected faults (for \( APFD \) and \( NAPFD \)) or the total severities of newly detected faults (for \( APFD_C \) and \( NAPFD_C \)) grow linearly. Taking the prioritized test suite \( \sigma_1 : T_3 - T_5 - T_2 - T_4 - T_1 \) as an example, in the scenario where both test costs and fault severities are ignored, there is a continue function, which from the number of test cases run (or consumed time) to the percent of faults detected, reflecting the process that \( \sigma_1 \) detects faults (see the curve in Fig. 1).

But factually, if a test case is still running, it cannot detect any faults, since it is impossible to check whether this test case is passed or failed before the end of running. It means that the function, which from the number of test cases run (or consumed time) to the percent of faults detected (or percent of total severities of detected faults), should be a step function in order to reflect the process of detecting faults. Also taking \( \sigma_1 : T_3 - T_5 - T_2 - T_4 - T_1 \) as an example, the corresponding step function is shown in Fig. 3. And the difference between continue function and step function is shown in Fig. 4.

So in computing mathematical expectation that explained in Theorem 1 for a prioritized test suite, there will be a margin of error. The margin of error may be very severe especially when the number of test cases is small. And if there are numerous test cases in prioritized test suite, we may accept such a approximation since the windage is minor.
V. IMPROVED METRICS

To avoid above limitations, we propose a series of improved metrics especially for non-classic test prioritization problems.

A. Relative-APFD

For two or more prioritized test suites that contain different number of test cases, if we want to evaluate and compare how rapidly they detect faults in the scenario where both test costs and fault severities are ignored, a fair testing resource should be provided firstly. Here the testing resource, which could be described as a positive integer $m$, is considered as a constraint:

1) $m < |\sigma|$: at most $m$ test cases in $\sigma$ will run.
2) $m \geq |\sigma|$: all the $|\sigma|$ test cases in $\sigma$ will run.

By using the testing resource constraint, we propose an improved metric called relative average percent of faults detected (Relative-APFD or RAPFD for short). The value of RAPFD does not only depend on the test suites under evaluation, but also relate to the given testing resource constraint. And further, this metric could handle non-detected faults.

Formally, let $\sigma$ be a prioritized test suite under evaluation, $\Phi$ the set of fault contained in the software, and $TF(\phi, \sigma)$ the position of the first test case in $\sigma$ that exposes fault $\phi \in \Phi$ ($TF(\phi, \sigma) = 0$ for non-detected fault). And let $C_i$ be the cost of the $i$-th test case ($i = 1, 2, ..., |\sigma|$), $S_\phi$ the severity of the fault $\phi \in \Phi$. For a given testing resource constraint $m$, the RAPFD of $\sigma$ is defined as:

$$RAPFD(\sigma, m) = p_m - \frac{\sum_{\phi \in \Phi} RTF(\phi, \sigma, m)}{m \times |\Phi|}$$  (6)

Where

$$RTF(\phi, \sigma, m) = \begin{cases} TF(\phi, \sigma) : m \geq TF(\phi, \sigma) \\ 0 : m < TF(\phi, \sigma) \end{cases}$$

And $p_m$ is the ratio of the number of faults detected by first $m$ test cases in $\sigma$ to the number of faults in $\Phi$:

$$p_m = \frac{|\{\phi \in \Phi | RTF(\phi, \sigma, m) \neq 0\}|}{|\Phi|}$$

Taking the prioritized test suite $\sigma_1 : T_3 - T_5 - T_2 - T_4 - T_1$ as an example, for a given testing resource constraint $m$, RAPFD($\sigma_1, m$) shows the area surrounded by the x-axis, y-axis, the line with $x = m$, and the curve that reflects the step function of fault detection process of $\sigma_1$. (See Fig. 5).

B. Relative-APFDCW

Considering the scenarios that test costs and fault severities are varying, the given uniform testing resource constraint should be scaled by a positive real number $m_c$, which indicates that the consumed testing resource should be less or equal to $m_c$. Then we can propose the metric called relative cost-cognizant weighted average percent of faults detected (Relative-APFDCW or RAPFDCW for short).

Formally, let $\sigma$ be a prioritized test suite under evaluation, $\Phi$ the set of fault contained in the software, and $TF(\phi, \sigma)$ the position of the first test case in $\sigma$ that exposes fault $\phi \in \Phi$ ($TF(\phi, \sigma) = 0$ for non-detected fault). And let $C_i$ be the cost of the $i$-th test case ($i = 1, 2, ..., |\sigma|$), $S_\phi$ the severity of the fault $\phi \in \Phi$. For a given testing resource constraint $m_c$, the RAPFDCW of $\sigma$ is defined as:

$$RAPFDCW(\sigma, m_c) = p_{mcw} - \frac{\sum_{\phi \in \Phi} S_\phi \times \sum_{i=1}^{RTF_C(\phi, \sigma, m_c)} C_i}{m_c \times \sum_{\phi \in \Phi} S_\phi}$$  (7)

Where

$$RTF_C(\phi, \sigma, m_c) = \begin{cases} TF(\phi, \sigma) : m_c \geq \sum_{i=1}^{RTF(\phi, \sigma)} C_i \\ 0 : m_c < \sum_{i=1}^{RTF(\phi, \sigma)} C_i \end{cases}$$

And $p_{mcw}$ is the ratio of the total severities of faults detected by $\sigma$ within the testing resource constraint, to the total severities of all faults in $\Phi$:

$$p_{mcw} = \frac{\sum_{\phi \in \Phi} S_\phi \times \sum_{i=1}^{RTF_C(\phi, \sigma, m_c)} C_i}{\sum_{\phi \in \Phi} S_\phi}$$

It is clear that RAPFDCW is equivalent to RAPFD when both test costs and fault severities are uniform.

And further, for the scenario where only test costs / only fault severities are taken into consideration, we can utilize

$$RAPFDC(\sigma, m_c) = p_{mc} - \frac{\sum_{\phi \in \Phi} \sum_{i=1}^{RTF_C(\phi, \sigma, m_c)} C_i}{m_c \times |\Phi|}$$  (8)

And

$$RAPFDW(\sigma, m) = p_{mw} - \frac{\sum_{\phi \in \Phi} S_\phi \times \sum_{i=1}^{RTF(\phi, \sigma, m)} C_i}{m \times \sum_{\phi \in \Phi} S_\phi}$$  (9)
respectively. Where

\[ p_{mc} = \frac{|\{\phi \in \Phi | RTF_C(\phi, \sigma, m_c) \neq 0\}|}{|\Phi|} \]

\[ p_{mw} = \frac{\sum_{\phi \in \Phi} RTF(\phi, \sigma, m_w) S_\phi}{\sum_{\phi \in \Phi} S_\phi} \]

They are both special cases of \( RAPFD_{CW} \)

C. Physical Explanation

We take the \( RAPFD_{CW} \) as example to analyze the physical meaning of improved metrics. \( RAPFD, RAPFD_C \) and \( RAPFD_W \) will be omitted, since they could be considered as a special case of \( RAPFD_{CW} \).

**Theorem 2.** Let \( C_i \) be the consumed time of the \( i \)-th test case \( T_i \in \sigma (i = 1, 2, ..., |\sigma|) \), \( S_\phi \) the severity of fault \( \phi \in \Phi \). Suppose the variable \( t \in [0, m_c] \) be the time of the moment testing process terminates exceptionally. The \( RAPFD_{CW}(\sigma, m_c) \) is equal to the mathematical expectation (or expected value) of the percent of total severities of faults detected before the termination, if following two assumptions hold:

1) \( t \sim U[0, m_c] \). It means that \( t \) follows the continuous uniform distribution with parameters \((0, m_c)\).

2) During the running of the \( i \)-th test case \( T_i \in \sigma \), the total severities of its newly detected faults keep still, until the execution of \( T_i \) is finished.

Proof is omitted since the length constraint.

The theorem means that proposed \( RAPFD, RAPFD_C, RAPFD_W \), and \( RAPFD_{CW} \) could help people to control the risk of testing process too.

VI. Examples

We still take the test cases and faults that shown in Table 1 as examples to illustrate the advantage of proposed \( RAPFD \).

Considering \( \sigma_1 : T_3 - T_4 - T_2 - T_4 - T_1 \) and \( \sigma_2 : T_3 - T_5 - T_6 \). For testing resource constraint \( m = 1, 2, 3, 4, \) and 5 respectively, we compute \( RAPFD \) for \( \sigma_1 \) and \( \sigma_2 \) (also see the area under the step functions in Fig. 5 and Fig. 6):

1) \( RAPFD(\sigma_1, 1) = RAPFD(\sigma_2, 1) = 0 \)
2) \( RAPFD(\sigma_1, 2) = RAPFD(\sigma_2, 2) = \frac{2}{5} \)
3) \( RAPFD(\sigma_1, 3) = RAPFD(\sigma_2, 3) = \frac{1}{3} \)
4) \( RAPFD(\sigma_1, 4) = \frac{13}{15} < RAPFD(\sigma_2, 4) = \frac{1}{2} \)
5) \( RAPFD(\sigma_1, 5) = \frac{2}{5} < RAPFD(\sigma_2, 5) = \frac{3}{5} \)

The overall results show us that: if testing resource constraint is greater than 3 (more than test cases could run), \( \sigma_2 \) detects faults more rapidly than \( \sigma_1 \); if testing resource constraint is less than 3 (less than 3 test cases could run), \( \sigma_1 \) and \( \sigma_2 \) have the same efficiency. And if testing resource constraint is 3 (just 3 test cases could run), though \( RAPFD(\sigma_1, 3) = RAPFD(\sigma_2, 3) \), \( \sigma_2 \) is more efficient than \( \sigma_1 \) since the former detects more faults using first 3 test cases.

And for other two prioritized test suites \( \sigma_3 : T_3 - T_2 - T_5 \) and \( \sigma_4 : T_3 - T_5 \), their \( RAPFD \) values for testing resource constraint \( m = 1, 2, 3 \) are:

1) \( RAPFD(\sigma_3, 1) = RAPFD(\sigma_4, 1) = 0 \)
2) \( RAPFD(\sigma_3, 2) = RAPFD(\sigma_4, 2) = \frac{3}{10} \)
3) \( RAPFD(\sigma_3, 3) = \frac{6}{15} < RAPFD(\sigma_4, 3) = \frac{8}{15} \)

The overall results show us that: if testing resource constraint is greater than 2, \( \sigma_4 \) detects faults more rapidly than \( \sigma_3 \); if testing resource constraint is less than 2, \( \sigma_3 \) and \( \sigma_4 \) have the same efficiency. And if testing resource constraint is 2, though \( RAPFD(\sigma_3, 2) = RAPFD(\sigma_4, 2) \), \( \sigma_4 \) is more efficient than \( \sigma_3 \) since the former detects more faults using first 2 test cases.

\( RAPFD_C, RAPFD_W, \) and \( RAPFD_{CW} \) have the similar advantage, which is omitted here.

VII. Conclusion

We make a brief revisit of widely used existing \( APFD \) metrics, discuss the their physical explanations, and point out some limitations that may lead incorrect results especially in non-classic test case prioritization problems. To avoid limitations, a series of improved metrics are proposed in this paper. They could illustrate the process of faults detection in software testing more precisely and practically, and provide physical meaningful results to evaluate and compare the efficiency of prioritized test suites.

Besides the theoretical analysis and simple examples, more applications and case studies should be investigated in future works to examine the proposed metrics.

ACKNOWLEDGMENT

Supported by the National Natural Science Foundation of China (61300054), Natural Science Foundation of Jiangsu Province (BK20130879), Natural Science Foundation for Colleges & Universities of Jiangsu Province (13KJB520018).

REFERENCES