2013 IEEE 37th Annual Computer Software and Applications Conference

On the Gain of Measuring Test Case Prioritization*

Junpeng Lv, Beibei Yin and Kai-Yuan Cai
School of Automation Science and Electrical Engineering
Beihang University, Beijing 100191, China
Email: realljp@buaa.edu.cn

***Abstract*—Test case prioritization (TCP) techniques aim to schedule the order of regression test suite to maximize some properties, such as early fault detection. In order to measure the abilities of different TCP techniques for early fault detection, a metric named average percentage of faults detected (*APFD*) is widely adopted. In this paper, we analyze the metric *APFD* and explore the gain of measuring TCP techniques from a control theory viewpoint. Based on that, we propose a generalized metric for TCP. This new metric focuses on the gain of defining early fault detection and measuring TCP techniques for various needs in different evaluation scenarios. By adopting this new metric, not only flexibility can be guaranteed, but also explicit physical significance for the metric will be provided before evaluation.**

Keywords-regression testing; test case prioritization; software metric; software cybernetics

I. Introduction

Software systems evolve with tests and modifications to provide the required functionalities and satisfy ever-changing customer needs. However, the modifications on existing software systems might break previous verified functionalities and introduce some regression faults. Thus, regression testing is adopted to detect these faults. The simplest regression testing strategy is to rerun all the existing test cases from earlier versions which might be costly and less efficiency [1]. Therefore, software testers might want to order the test cases so that some test cases with higher priority, according to some criterion, are run earlier to improve the effectiveness of testing.

Test case prioritization (TCP) techniques [2-4] aim to schedule the order of test cases so that some objective function can be maximized. One potential goal of test case prioritization is to increase a test suite's rate of fault detection – that is, how quickly that test suite detects faults during the testing process [5].

On evaluating the performance of different TCP techniques, a metric named average percentage of faults detected (*APFD*) is proposed by Rothermel et al.[5] which aims to "measure the average cumulative percentage of faults detected over the course of executing the test cases in a test suite in a given order". *APFD* is shown to be applicable for qualifying and comparing the performance of different TCP techniques [6-9]. Moreover, a more general metric, *APFDc*, is proposed by incorporating varying test costs fault severities into test case prioritization [10].

In spite that *APFD* and *APFDc* have been widely adopted in TCP techniques, the gain of measuring TCP techniques with *APFD* series has not been explicitly discussed according to our knowledge. In control theory, the relationship between input and output is denoted by a transfer function as depicted in Fig. 1. The relationship between system output y and input r can be expressed as $y = \frac{G}{1+GH} \cdot r$, where G is called the open-loop gain which denotes the open-loop relationship between r and y. In control theory, the feedback controller H is introduced to guarantee the output y can be limited to admirable value even if the open-loop gain G is not so good. The gain of measuring TCP techniques with *APFD* series is similar with the control approach. *APFD* treats the test case execution and faults detection information as the input and transfers it into a single number as an output which can be mathematically considered as an open-loop gain. This kind of compression will lead to some loss of information and thus how the process is undertaken should be investigated. For example, in *APFD* series, one technique with higher *APFD* values is defined to have a "faster" fault detecting ability. However, the problem is how the "fastness" is defined? Is one technique detecting all known faults earlier always "faster" than another with more test cases? Thus, an explicit description on the definition of "fastness" in *APFD* is needed before adopting it in TCP researches.

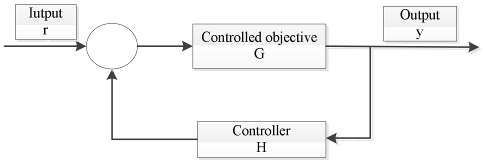


Figure 1 classic feedback loop in control theory

Base on the analysis on *APFD*, a new generalized metric on performance of TCP techniques, named weighted gain of faults detected (*WGFD*), is proposed in this paper. This new metric defines the "fastness"/gain in a direct way by weighting and summing the faults detection rates into a signal metric. Besides, it can also be extended to incorporate different needs into test case prioritization. Moreover, *WGFD* also allows testers to build their own definition of "fastness" more flexibly according to their requirement.

The rest part of this paper is organized as follows. Section II describes the background of TCP metrics, such as,

*This work is supported by the National Natural Science Foundation of China (Grant Number 61272164) and the Beijing Natural Science Foundation (Grant Number 4112033)

0730-3157/13 $26.00 © 2013 IEEE
DOI 10.1109/COMPSAC.2013.101

APFD series; "gain" analysis on *APFD* is done in section III; a more general metric *WGFD* is proposed in section IV to give a clear insight into the measurement for TCP techniques. Conclusion and future work are mentioned in Section V.

II. BACKGROUND

Regression testing is an important and expensive maintain process in software developing process. In order to improve the effectiveness and efficiency of regression testing, many techniques have been studied to improve the effectiveness and efficiency of regression testing, such as, regression test suite minimization, regression test case selection and regression test case prioritization [11-12].

On evaluating the performance of different TCP techniques, *APFD* is proposed [5]. The basic idea of *APFD* is to measure a prioritized test suite's efficacy and average fault detection, which is "analogous to measuring an antibiotic's potency and average activity". The values of *APFD* range from 0 to 100, but usually it is not a tight bound [13], which means some values are not available in the range; and higher *APFD* values mean faster fault detection rates.

To illustrate the metric, an example program with 10 faulty versions and a set of five test cases are given [5]. Table I shows the fault detecting ability of these test cases.

Table I TEST SUITE AND LIST OF FAULTS EXPOSED

Test Case	Fault									
	1	2	3	4	5	6	7	8	9	10
A	X				X					
B	X				X	X	X			
C	X	X	X	X	X	X	X			
D					X					
E								X	X	X

Three prioritized test suites are constructed with three different orders of five test cases. Test cases are replaced with the order A-B-C-D-E in test suite T_1, E-D-C-B-A in T_2 and C-E-B-A-D in T_3. Fig. 2 shows the percentage of detected faults versus the fraction of the test suite used for all three test suites. Clearly, T_3 detects all the faults with 0.4 of the whole test suite and therefore, it has the fastest fault detecting rates as well as the largest *APFD* value of 84% among three test suites.

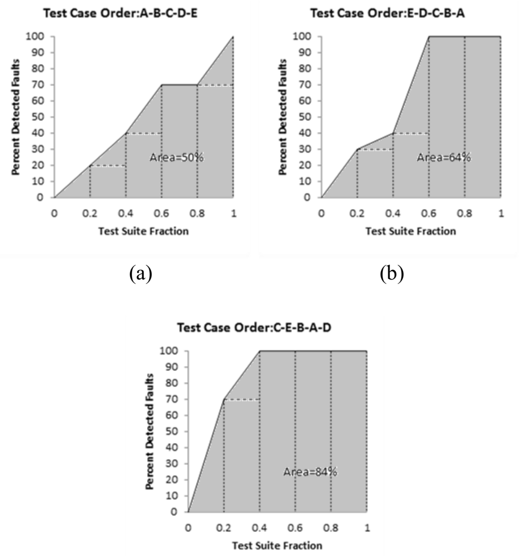
Let T be the test suite containing n test cases and let F be the set of m faults revealed by T. For ordering T, let TF_i be the order of the first test case that reveals the ith fault. According to the definition of *APFD*, the area under the curve can be calculated as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots + TF_m}{nm} + \frac{1}{2n} = \frac{\sum_{i=1}^{m}(n - TF_i + \frac{1}{2})}{nm} \quad (1)$$

In order to incorporate varying test costs and fault severities into TCP, an improved metric $APFD_C$ is proposed [10]. This $APFD_C$ considers "the tradeoff between testing cost and the costs leaving undetected faults in software", thus this metric focuses on "rewarding test case orders proportionally to their rate of "units-of-fault-severity-detected-per-unit-test-cost"" [10]. The following equation shows the calculation of $APFD_C$:

$$APFD_C = \frac{\sum_{i=1}^{m}(f_i \times (\sum_{j=TF_i}^{n} t_i - \frac{1}{2} t_{TF_i}))}{\sum_{i=1}^{n} t_i \times \sum_{k=1}^{m} f_k} \quad (2)$$

where t_i denotes the testing cost of ith test cases, and f_k denotes the fault severity of the kth detected fault.



(a) (b)

(c)

Figure 2. An example illustrating *APFD*. (a) *APFD* for priorized test suite T_1. (b) *APFD* for priorized test suite T_2. (c) *APFD* for priorized test suite T_3.

III. ANALYSIS ON *APFD*

APFD and $APFD_C$ have been widely used in researches on TCP techniques and some other transformations on *APFD* are proposed [4]. However, according to our knowledge, there is still some confusion on current *APFD* metric series:

A. On the gain of APFD

1) Analogous to empirical pharmacodynamic indices of antibiotic effectiveness, the *AFPD* for TCP measurement is similar to the area under curve (*AUC*) in noncompartmental pharmacokinetics analysis. The *AUC* curve denotes the drug concentration, such as, that in serum; whereas the *APFD* curve denotes the number of cumulative detected faults. This will lead to a problem: the physical significance of *AUC* can be explained as the evaluation of total drug exposure, that is, how much and how long a drug stays in a body; then, what are the physical significance of *APFD*? According to (1), *APFD* has a negative correlation with the sum of fault detection cost and the detection cost for each test case is negatively proportional to its detection rate. Thus, it indeed measures how rapidly a prioritized test suite detects faults.

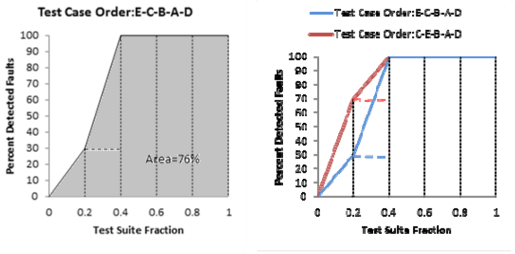
Note that, there are various ways to combine the fault detection rates into a single number, and *APFD* is just one of them. Do all these ways have the same effect on measuring the performance of different TCP techniques? If no, it is necessary to mention the way or the gain of processing the vector into the single number before adopting it.

First, let's consider the compensatory part of *APFD* in the graph, that is the area upon the fault detection curve, or the area between the curve and the line y=100. The physical significance of that area is the total fault survival "time", and thus it can be named average percentage of faults survival (*APFS*),. It is natural to believe that less fault survival "time" denotes faster fault detection, and thus *APFS* is a decreasing function of early fault detection. In this way, the "fastness" of one technique can also be defined negatively correlated to the area upon the curve. Note that, $APFD = 1 - APFS$. Due to these two different approaches to define the "fastness" of fault detection, the results may be quite different when measuring the same test suite.

An example is given to illustrate this difference using the example program in Table I. First, a new prioritized test suite T_4 with test case order E-C-B-A-D is given. The *APFD* value for T_4 is calculated in Fig. 3a. Comparing the two curves for T_3 and T_4 in Fig. 3b, it is definite that test suite T_3 detects faults faster than T_4, and the *APFD* value of T_3 is also higher than that of T_4, which is 84% to 76%. According to the *APFD* values, the advantage of T_3 over T_4 is 8%. Moreover, the *APFS*s for T_3 and T_4 are 16% and 24% separately with also the deviation of 8%. Note that *APFD* is an increasing function of early fault detection but *APFS* is a decreasing one, thus both metrics indicate T_3 is better than T_4. Note that, *APFD* values are not the truly normalized values for each test suite to fulfill [0,100] [13], thus the 8% is not the deviation in true ranking position for T_3 and T_4. Under the situation with true ranking unknown, the possible way to make decision on which how much improvement can T_3 provide over T_4 is to evaluate their relative difference. The relative difference between T_3 and T_4 in *APFS*, that is, $8\% / 24\% \times 100\% = 33.3\%$, seems much larger than in *APFD*, that is $8\% / 84\% \times 100\% = 9.5\%$, as in *APFS* the area is much smaller than in *APFD*. In this case, the results can be confusing when comparing two test suites. According to *APFD*, T_3 improves only 9.5% over T_4 and if the cost of T_3 is much larger than that of T_4, it may not be suitable to adopt T_3 with some limited budget, whereas the conclusion provided by *APFS* can be opposite as T_3 improves 33.3% over T_4.

As we can see, when measuring the performance of TCP techniques, the relative ability of detecting faults between two techniques might vary a lot with different metrics. That is, when the absolute values of deviation between two TCP techniques are the same, but their values are different. The problem is, if these two metrics are both reasonable and the relative difference between the fault detection abilities of two techniques are as large as *APFS* denotes or as small as *APFD* denotes, what should we do? The different results on *APFD* and *APFS* indicate that the measurement of TCP techniques are no unique, and before adopting any metric on that, the gain of that metric should be explicitly described.

The definition of *AUC* denotes that a long, low concentration exposure may be as important as shorter but higher concentration. Thus, there should be a quantified relationship between with exposures with different length and height, and so should *APFD* be.



(a) (b)

Figure 3. An example illustrating *APFD* involving areas after all faults are detected. (a) *APFD* for test suite T_4. (b) Faults detecting ability comparison between T_3 and T_4.

Equation (1) shows the calculation of *APFD*. As the calculation is processed based on the *faults*, in order to investigate the explicit definition of "fastness" for each *test case* in *APFD*, we just assume the *j*th and *k*th test case detect N_j and N_k faults separately in some prioritized test suite T_x. Now suppose another test suite T_y detects N_j' and N_k' faults separately with the *j*th and *k*th test cases. The procedures for T_x and T_y detecting other faults are just the same. And coincidently the *APFD* values for two test suites are the same. It means that

$$N_j(n-j+\frac{1}{2})+N_k(n-k+\frac{1}{2})=N_j'(n-j+\frac{1}{2})+N_k'(n-k+\frac{1}{2}) \tag{3}$$

Thus, a quantified relationship in *APFD* is obtained by simple manipulation

$$\frac{\Delta N_j}{\Delta N_k}=\frac{N_j'-N_j}{N_k'-N_k}=-\frac{n-k+\frac{1}{2}}{n-j+\frac{1}{2}} \tag{4}$$

Similarly, the quantified relationship in *APFS* can be obtained as follows

$$\frac{\Delta N_j}{\Delta N_k}=\frac{N_j'-N_j}{N_k'-N_k}=-\frac{k-\frac{1}{2}}{j-\frac{1}{2}} \tag{5}$$

Note that the negative sign denotes that if one test case detects fewer faults then the other needs to detect more for compensation. The existence of (4) illustrates an explicit definition of "fastness"/gain in *APFD*. It implies the effect of earlier detected fault and gives a quantified compensatory relationship for detecting faults with different orders of test cases. Note that, in *APFS*, if one test case detects fewer faults than thought, the total effect on the *APFS* is the product of number of reduced faults and almost the order of the test case, which can be considered as more faults survival. The 1/2 in (4) and (5) is just the result of using a straight line to connect the various points on the curve as discussed in [13].

Comparing (4) with (5) we can see that, when only the initial test cases are considered, the contribution per fault to the *APFD* value for each test case does not vary a lot especially when n is large, whereas it does not hold for *APFS*. This difference is caused by the involvement of n in calculation for *APFD* and such involvement leads to less insensitivity to the fault detection rate changes compared with *APFS*. This can explain the difference between relative improvements over T_4 with two metrics for measuring T_3.

Thus, an important thing to mention before adopting a metric is to announce the gain for it. Different gain will lead to different physical significance. Whether a metric has some intense tendency on some properties, such as, early fault detection in initial test cases, should be explicitly mentioned before making any decisions. For example, in safety-critical systems, the evaluation of TCP techniques may be much "critical" as the earlier fault detection may contribute more. Thus, in this case, the metric *APFS* seems to be much more suitable to making decisions than *APFD*.

B. *On the range of APFD*

According to Zhang and Qu [13], *APFD* is not a complete normalized metric. It is claimed that the value for *APFD* ranges from 0 to 100, but the actual interval is from $1/(2n)$ to $1-1/(2n)$, which is also not true as if no faults are detected then then *APFD* is 0. Thus, the interval is from 0 to $1-1/(2n)$. Moreover, usually the interval cannot be fulfilled, that is, the boundaries are not tight. Therefore, the value of *APFD* is not capable of determining "quality of the ordered test suite" or its true ranking position in all possible results. Thus, an improved *APFD* metric, $AFPD_b$, is proposed by normalizing the *APFD* value considering its actual value range for each test suite.

The normalization issue concerned in [13] can be illustrated by the following example. It is obvious that the maximum of *APFD* for all prioritized test suite in the example shown by Table I is 84%, which is the *APFD* of T_3, and the minimum is 38% with the test case order D-A-B-C-E or D-A-B-E-C. When considering another test suite shown in Table II for the same program, the maximum and minimum are different. In this case, the maximum of *APFD* is 74% with the order C-E-A/D-D/A-B and the minimum is 48% with the order D-A-B-C-E. In this scenario, it is not feasible to comparing the performance of the same TCP technique on different test suite for same subject program by *APFD*. As the different test suite may have different boundary values for *APFD*, thus the previous *APFD* value does not represent the absolute or relative fault detecting ability of TCP technique. In this case, if these two maximum value are achieved by same TCP technique and they are plotted in the same boxplot, then then value 74% will be possibly treated differently with the 84% but actually they should be treated equally as they both reach the upper bound of the test suite. As the range of *AFPD* is test suite dependent and the purpose of using *APFD* is to investigate the improvement on original test suite taken by the TCP technique, it is rational to normalize the *APFD* value for different test suite before adding it into a box plot.

Table II ANOTHER TEST SUITE AND LIST OF FAULTS EXPOSED

Test Case	Fault									
	1	2	3	4	5	6	7	8	9	10
A	X	X			X					
B	X				X	X	X			
C	X			X	X	X	X			
D			X		X					
E								X	X	X

As the range of *APFD* is needed to calculate the $AFPD_b$, Zhang and Qu propose an approximate approach for estimating the range; whereas, the approximation is based on the assumption that each test case only detects one fault. This assumption limits the utilization of their conclusion as in many scenarios one test case many detect more than one fault. Besides, some greedy algorithm and search based techniques adopted in [1][4] can also be revised to provide approximated values. But one problem is that these algorithms may run into a suboptimal solution and the deviation between a suboptimal solution and global optima may vary a lot. These two approaches cannot guarantee the precision of range calculation and thus the approach of normalizing the relative value is not always available.

The precision of approximation on the range of *APFD* will restrict the accuracy of determining the performance of different TCP techniques. Another threat to the effectiveness of $AFPD_b$ is that even though the range of *APFD* value is precisely calculated, the range cannot be fulfilled as the permutation of a test suite is limited, that is, that value of *APFD* for some test suite is not continuous but discrete. This means that the way that $AFPD_b$ locates the performance of one technique is still not accurate enough as some value in this range may not available for this test suite. For example, it may occur that in test suite I, one result that is not the worst cannot be less than 50%, but in another test suite II it is possible but not larger than 50% except for the upper bound. Thus, comparing results in same boxplot for same TCP techniques may not be suitable as it may occur that the second best performance in test suite II get a lower value than the second worst performance in test suite I. it is not fair to compare the performance of the TCP techniques in such a way. Thus one possible accurate solution should be listing all the possible value of *APFD* for the same test suite, sorting them into a queue and locating the result of some technique by its position in the queue in percentage. But this approach is too time consuming and not likely applicable.

Note that, this section mainly focuses on the confusion in *APFD*. Actually, these problems lie in all the *APFD* series. Due to different definition of the gain, the results of *APFD* and *APFS* might diverse a lot in relative deviation of fault detection ability. Another difference between *AUC* and *APFD*/*APFS* is that the *AUC* is an absolute value while *APFD*/*APFS* is a relative one by dividing by the product of number of test case and fault. As the *APFD* series has such relative value, it is reasonable to renormalize the value to obtain *AFPD*$_b$. However, in order to get the range of *APFD*, some approximation should be adopted which will affect the result of normalization. Besides, due to the limited permutation of a test suite, the normalization in *AFPD*$_b$ can still produce some error as the range is not continuous but discrete.

In practice, testers may suffer to different extent from the fault detection ability, thus a fixed compensatory relationship is not enough to meet different needs. And building up a metric with flexible and easy-understanding gains according to the tester's own willing is considerable. When considering the "fastness" without a clear declaration, it seems the results may vary a lot with different metrics. Thus, it is natural to explore the purpose of measuring different TCP techniques in different testing scenarios and define a generalize metric with explicit gains and physical significance.

IV. A GENERALIZED METRIC FOR TCP TECHNIQUES

In this study, a generalized absolute metric is proposed similar with the adoption *AUC*. Similarly with *AUC*, this new metric focuses on the absolute value of fault detection ability as well as flexibility in adoption.

A. A generalized metric

This section proposes a new flexible and easy-understanding metric for measuring the fault detecting ability. The basic idea is to weight and sum *fault detecting rates* of different test cases to build a metric that can represent the fault detecting ability of entire test suite. Note that, the fault detection rate for each test case is calculated as follows

$$r(i) = \frac{f(i)}{i-(i-1)} = f(i) \tag{6}$$

where *r*(*i*) denotes the fault detecting rate for the *i*th test case and *f*(*i*) denotes the number of fault detected by the *i*th test case. Considering the number of test cases detected at different time should have different impact on measuring the fastness of a TCP technique, the a weight *w*(*i*) should be assigned to *r*(*i*). Before weighting the fault detecting rates of different test cases, the basic rules for measuring a "fastness" of TCP techniques should be emphasized:

- If $i < j$ and $f(i) = f(j)$, then the weighted value of impact for test case *i* and *j* should follow that $w(i)\bullet f(i) > w(j)\bullet f(j)$, which means the if same number of faults are detected earlier, the metric should be larger and thus the technique should be considered faster. This determines a metric positively correlated to the fault detecting rate and surely a negatively correlated one can also be defined similarly. In the following part, we take the positively correlated metric as example.
- According to investigation on *APFD*/*APFS*, if the metric is dependent on the test suite, the range issue should be solved anyway without no guarantee on the precision of the solution. Thus, in this metric, the weight *w*(*i*) should be a constant or a function of *i* but not supposed to be function of test case numbers or some other test suite dependent properties. This provides the condition that guarantees the uniqueness for determining a metric after determining the weights. Although flexibility is promised in this metric, the weights and the metric should be independent of properties of test suites to avoid any possible bias. And the flexibility should lie in the choosing the weights for test cases according to different needs but not different test suites.
- If $i < j$ and $f(i) < f(j)$, then how the weights are assigned should be pointed out explicitly as this implies the priorities assigned to test case order and detecting fault number. Or in a simplified way, how many more test cases detected later can be considered to compensate the loss of not doing so earlier? This relationship in *APFD* is denoted by (4) and it is related to the number of test suite, which is not preferred in this study. As mentioned above, the weights should be independent of such test suite properties to avoid bias.

Keeping the above three rules in mind, a generalized weighted metric for measuring the rates of fault detection of different TCP techniques, namely weighted gain of fault detection (*WGFD*) can be established as follows:

$$WGFD = \sum\nolimits_{i=1}^{n} w(i)\bullet r(i) \tag{7}$$

B. Instances of WGFD

The *WGFD* propose basic rules to measuring the fault detecting ability of TCP techniques. Note that, the weights are just abstract but with no implementation. In order illustrate the *WGFD*, some instances are shown in this section.

1) Instance type I: One possible consideration is that the weight of *i*th test case is just reversely proportional to the test order *i*. Then the weights can be rewritten as $w(i) = c_i / i$, where c$_i$ is a constant which denotes the relative priority for test case i excluding the effect of test order. These constants can be equal or not, depending on the choice of measuring the TCP techniques. Thus, the *WGFD* can be instantiated as

$$WGFD_I = \sum\nolimits_{i=1}^{n} \frac{c_i}{i}\bullet r(i) = \sum\nolimits_{i=1}^{n} \frac{c_i}{i}\bullet f(i) \tag{8}$$

2) Instance type II: Another choice of weights can be a sequence of number somewhat negatively correlated to the position i, which denotes a scenario that the weights for different position is much less skewed than in (8). Thus, an negatively correlated *WGFD* can be given as follows

$$WRFD_{II} = \sum\nolimits_{i=1}^{n} (a - b\cdot i)\bullet r(i) = \sum\nolimits_{i=1}^{n} (a - b\cdot i)\bullet f(i) \tag{9}$$

Note that, in this *WGFD*$_{II}$ if $a = (n+1)/(2nm)$ and $b = 1/(nm)$, the *WGFD*$_{II}$ is exactly proportional to the *APFD*. And if $a = 1/(2nm)$ and $b = -1/(nm)$, the *WGFD*$_{II}$ is exactly proportional to the *APFS*. However, in this scenario, the

$WGFD_{II}$ is negatively correlated to the rates of fault detection. However, as mentioned in the concerns on *APFD*, the test case number *n* will lead to unavoidable range computation whose precision is not guaranteed.

C. Extension for WGFD

In order to incorporate different needs into *APFD*, many metrics are proposed [4, 9, 14]. The *WGFD* in this study can also be incorporated the varying test cost and fault severity into. Considering similar ways as $APFD_c$, the rate of fault detection for each test case actually transforms into the ratio of fault severity to test cost. Thus the $r(i)$ in (9) transforms into $rs(i)$, which denotes the rate of fault survival severity

$$rs(i) = \frac{fs(i)}{tc(i) - tc(i-1)} \tag{10}$$

where $fs(i)$ denotes the fault severity exposed by test case *i* and $tc(i)$ denotes the total test cost after test case *i* has been executed. And in this scenario, the *WGFD* transforms into

$$WGFD = \sum_{i=1}^{n} w(i) \bullet rs(i) = \sum_{i=1}^{n} \frac{w(i) \bullet fs(i)}{tc(i) - tc(i-1)} \tag{11}$$

The weights selection is similar with the (8) and (11), or according to the developers' needs.

In this section, a new generalized metric *WGFD* is proposed to measure the performance of TCP techniques based on weighting and summing the fault detection rate for each test case. The *WGFD* is an absolute metric which has explicit physical significance and a flexible implementation approach. In this way, *WGFD* can provide measurement on the performances of TCP techniques on different test suites. Besides, the flexibility of *WGFD* also makes it more adoptable in many scenarios.

V. Conclusion and Future Work

This study investigates the popular metric *APFD* and its extensions in TCP researches. The basic idea of *APFD* is to measure a prioritized test suite's efficacy and average fault detection. However, there are some confusion on *APFD*.

1) Due to the variety in defining the "fastness" of fault detection, such as *APFD* and *APFS*. Thus, it is important to announce the physical significance of metric before adopting it to give explicit picture comparison.

2) Besides, the *AUC* is an absolute value metric while *APFD* is with some unifing process which makes it closer to but not a relative metric independent of test suite. In order to otain the relative metric, the range of *APFD* for different test suite needs calculation whereas the precision of calculation is not guaranteed.

3) Moreover, the fixed gain/"fastness" in *APFD*/*APFS* cannot account for all the testing scenarios with different needs of measuring the fault detection ability.

In this study, a generalized metric *WGFD* is proposed. This new metric is constructed based on weighting and summing the rate of fault detection for each test case to explicitly denote the gain and physical significance of itself. Moreover, as the *WGFD* is an abstract metric which means the implementation can be adjusted, it is flexible for researchers to give a comparison between different TCP techniques with an explicit physical significance and parameter tuning for different scenarios. Also, two instances are given along with their physical significance to show an example of this new *WGFD*. Besides, this *WGFD* can also be extended to adapt to varying test cost and fault severity which makes it more potential to be utilized in extensive scenarios. The future work should include utilizing *WGFD* in TCP techniques comparison and investigation on the favorite application environments for different *WGFD* metrics.

References

[1]. G. Rothermel, R. H. Untch, C. Y. Chu, and M. J. Harrold, "Prioritizing Test Cases For Regression Testing", *IEEE Transactions on Software Engineering*, vol. 27(10), 2001, pp. 929-948.

[2]. Md. J. Arafeen, and H. Do, "Adaptive Regression Testing Strategy: An Empirical Study", in *Proceedings of 22nd IEEE International Symposium on Software Reliability Engineering*, Hiroshima Japan, 2011, pp.130-139.

[3]. S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies", *IEEE Transactions on Software Engineering*, vol. 28(2), 2002, pp. 159-182.

[4]. Z. Li, M. Harman, and R. M. Hierons, "Search Algorithms for Regression Test Case Prioritisation", *IEEE Transactions on Software Engineering*, vol. 33(4), 2007, pp. 225-237.

[5]. G. Rothermel, R. H. Untch, C. Y Chu, and M. J. Harrold, "Test Case Prioritization: An Empirical Study", in *Proceedings of International Conference on Software Maintenance* (*ICSM'99*), Oxford, UK, 1999. pp. 179-188.

[6]. S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey", *Journal of Software Testing, Verification and Reliability*, vol. 22, 2012, pp. 67-120.

[7]. S. Mirarab and L. Tahvildari , "An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization", in *Proceedings of the 1st International Conference on Software Testing, Verification, and Validation*, Lillehammer Norway, 2008, pp. 278-287.

[8]. H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel, "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments", *IEEE Transactions on Software Engineering*, vol. 38(5), 2010, pp. 593-617.

[9]. L. Tahat, B. Korel, M. Harman and H. Ural, "Regression Test Suite Prioritization Using System Models", *Journal of Software Testing, Verification and Reliability*, vol. 22, 2012, pp. 481-506

[10]. S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization", in *Proceedings of the International Conference on Software Engineering* (*ICSE 2001*), Toronto Canada, 2001, pp. 329-338.

[11]. H. Do, G. Rothermel, and A. Kinneer, "Prioritizing JUnit Test Cases: An Empirical Assessment and Cost-Benifit Analysis", Emperical Software Engineering, vol. 11, 2006, pp. 33-70.

[12]. M. J. Harrold. "Testing evolving software", *Journal of Systems and Software*, vol. 47(2–3), 1999, pp. 173–181.

[13]. X. F. Zhang and B. Qu, "An Improved Metric for Test Case Prioritization", in *Proceedings of 2011 Eighth Web Information Systems and Applications Conference*, Chongqing, China, 2011, pp. 125-130.

[14]. B. Qu, C. H. Nie, B. W Xu, and X. F. Zhang, "Test Case Prioritization for Black Box Testing", in *Proceedings of Computer Software and Applications Conference*(COMPSAC 2007), Beijing China, 2007, pp. 465-474.

[15]. K. Y. Cai, J. W. Cangussu, R. A. DeCarlo, and A. P. Mathur "An Overview of Software Cybernetics", in *Proceedings of the 11th Annual International Workshop on Software Technology and Engineering Practice* (STEP'03), Amsterdam Netherland, 2003, pp. 77-86.