A Weight-based Approach to Combinatorial Test Generation

Jing Zhao
Dalian University of Technology
China
zhaoj9988@dlut.edu.cn

G.R Ning
The Fourth Academy of CASIC
China
ninggaorong@buaa.edu.cn

H.L Lu
Harbin Engineering University
China
n.betula.lu@gmail.com

Y.B Wang
Harbin institute of Technology
China
wangyb@hit.edu.cn

Yan Cai
Institute of Software, Chinese
Academy of Sciences
China
y-cai.mail@gmail.com

Jian Zhang
Institute of Software, Chinese
Academy of Sciences
China
zj@ios.ac.cn

ABSTRACT

Combinatorial testing (CT) is very efficient to test parameterized systems. Kuhn et al. investigated the interaction faults of some real programs, and found that the faulty combinations are caused by the combination of no more than 6 parameters. Three or fewer parameters triggered a total of almost 90% of the failures in the application[3]. However, for high-quality software, simply testing all 3-way combinations is not sufficient [5], which may increase the risk of residual errors that lead to system failures and security weakness[4]. In addition, the number of test cases at 100% coverage for high-way is huge, which is beyond the farthest test overhead restrictions. *Covering array* is typically used as the test suite in CT, which should convey much information for the fault detection. We firstly proposed a weighted combinatorial coverage (CC), focusing on the fault detection capability of each test case instead of 100% percent t-way CC. Secondly, we give the test case selection algorithm FWA (fixed weight algorithm) using weighted CC metric. For generating each test case, our methodfi rst randomly generates several candidates, and selects the one that has the highest fault-detection possibility with the different sampling pool size. Thirdly, we give the theorems for our algorithm and definitions for the weighted CC. Finally, we compared the selected sample sized and the fault-detection capabilities of FWA as well as t-wise algorithms by using the four benchmarks with configuration options interaction faults, and we found FWA is able to detect higher number of faults with the less selected sample size, specifically, FWA is able to detect high-wise interaction faults with the less selected sample size compared with the 4-wise as well as 5-wise algorithms.

KEYWORDS

combinatorial testing; weighted combinatorial coverage; fault detection rate; test case selection;

ACM Reference Format:

Jing Zhao, G.R Ning, H.L Lu, Y.B Wang, Yan Cai, and Jian Zhang. 2018. A Weight-based Approach to Combinatorial Test Generation . In *ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3183440.3195018

ICSE '18 Companion, May 27-June 3, 2018, Gothenburg, Sweden

ACM ISBN 978-1-4503-5663-3/18/05.
https://doi.org/10.1145/3183440.3195018

1 INTRODUCTION

Software testing is a common way of ensuring software quality. Although software testing cannot guarantee the system under test (SUT) is error-free, a good testing practice can help detect many defects in the SUT, and provide higher confidence.

There are many software testing techniques, each of which is suiting for some circumstances. *Combinatorial testing* (CT) is one of these techniques,f which is very efficient to test parameterized systems [1, 2, 7, 8]. In CT, the SUT is modeled as a black-box, whose behavior is affected by several input parameters (or called parameter/configurations). Each parameter may take several possible values. From a black-box view, the failures are caused by the combination effects of parameters. These faults are called *interaction faults.* These failure-causing parameter combinations are called *faulty combinations.* Without any a prior knowledge, any parameter combination may be faulty. In the worst case, we need to test all possible parameter combinations, and the number of test cases increases exponentially with the increasing number of parameters. The number of i-way tests that will be required is proportional to $v^i \log n$, for n parameters with v values each [1, 4]. Kuhn et al. [3, 5, 6] investigated the interaction faults of some real programs, and found that the faulty combinations they studied are caused by the combination of no more than 6 parameters. Empirical investigations have concluded that from 70% to 90% of the failures could be identified by pairwise combinatorial testing[6]. This conclusion is also supported by other studies [10, 11]. If we have tested all the low-way combinations (e.g. all 3-way combinations), we can detect many interaction faults, and three or fewer parameters triggered a total of almost 90% of the failures in the application[3]. However, for high quality software, simply testing all 3-way combinations is not sufficient [5], which may increase the risk of residual errors that lead to system failures and security weakness[4].

Looking back to the typical way of conducting combinatorial testing, we give the covering strength in advance, specifying which combinations need to be covered (we call them *target combinations*), and generate a CA to cover all these combinations. Then the test generation is done. We use i-way combinatorial coverage as a metric to select the best test case. The chosen coverage criterion should be chosen or determined by the balance between test efficiency and cost. Suppose a Software Under Test(SUT) includes 20 parameters, and each parameter is out of 10 values, we can check all pairs of these values with only 180 tests by using pair-wise coverage, if N-way coverage used then 10^{20} test cases will be used, which is far more than what a software tester would conduct in a lifetime [7]. Forfi nding the remaining faults, one way is to change the t-wise coverage criterion. If we change the coverage criterion from pair-wise to 5-wise, Almost all the faults can be found with 100% coverage. Furthermore, if we employ 6-wise combinatorial coverage, the testing process will be forced to stop at a large SUT because the number of test cases at 100% coverage are huge, which is beyond the farthest test overhead restrictions. The problem is that we focus too much on the target combination.

In software testing, we should consider test effectiveness as well as efficiency for a SUT. And an efficient test case requires that the generated test case should contribute much useful information which can be used tofi nd faults in the SUT. We usually want to detect as many defects but with as less test effort as possible, and also to detect defects as early as possible. For this need, the fault-detection rate of the test suite is important. In CT, we usually use the covering array to represent a test case. Thus, a covering array should convey much information for the fault detection, and we should shift our focus from i-way combinatorial coverage to the fault-detection possibility of each test case. We measured the exposed average faults by executing each test case, in which the faults will be 1-way, 2-way,..., 6-way, etc. In Dr Kuhn works [5–7], the empirical reports have concluded faults interaction proportions between two-parameter values, three, four,fi ve and six-way. Therefore, we should concern the fault detection rate for the sum of i-way faults. Consequently, the combinatorial coverage can be regarded as the sum of the weighted i-way coverage.

Although combinatorial coverage is a good metric for guiding combinatorial testing, it includes the test case generation, selection, prioritization, etc. Traditionally t-way combinatorial coverage metric does not represent well the real defect distribution of t-way faults since those t-way faults from 1-way to 6-way co-exist in the real system, while t-way faults, such as pair-wise or higher-way fault interactions are just a special type of our weighted approach. Thus, our proposed combinatorial testing approach guided by weighted combinatorial coverage, focuses on generating the optimized test case that has the highest fault detection rate, instead of selecting at 100% percentage combinatorial coverage.

Our approach is based on the one-test-at-a-time strategy. To generate each test case, our methodfi rst randomly generates several candidates, and then selects the one that has the highest fault-detection possibility. The fault-detection ability is calculated based on the presumed distribution of faulty combination size, which could be given in advance based on one's experience and knowledge about the SUT. The traditional way of CT is a special case of our method where all faulty combinations are of size i, which requires choosing the covering strengthfi rst, if the strength is set too large that resources will be wasted or forced to stop, or if the strength is set too small that the covering array is insufficient tofi nd the specific failures[9].

In summary, our main contributions are as follows:

- We propose the metric of weighted combinatorial coverage, which connects the combinatorial coverage and the fault detection capability of each test case.
- We propose an algorithm FWA for the weighted combinatorial test case selection, and it is theoretically proved that the efficiency of FWA.
- We give the test efficiency comparisons with other t-wise algorithms using the four test benchmarks. sampling algorithms by conducting real-world benchmarks.

2 CONCLUSION

Combinatorial testing (CT) is very efficient to test parameterized systems. *Covering Array* is typically used as the test suite in CT, which should convey much information for the fault detection. Firstly, for generating each test case, our method randomly generates several candidates, and selects the one that has the highest fault-detection possibility using our proposed weighted combinatorial coverage. We give the theorem proof that our proposed weighted combinatorial coverage rate equals the fault detection rate. Secondly, we give the test case selection algorithm using weighted combinatorial coverage metric. And also, we give the theorem for our algorithm and definitions for the weighted combinatorial coverage. Finally, we conduct our experiments with four benchmarks, compared with five sampling algorithms in different metrics. The experimental results show that FWA has a better effectiveness and performance in detecting interaction faults.

REFERENCES

[1] D.M. Cohen, D.R. Dalal, M.L. Fredman, and G.C. Patton. 1997. The AETG system: an approach to testing based on combinatorial design. 23, 7 (1997), 437–444.

[2] M. Grindal, J. Offutt, and S.F. Andler. 2005. Combination Testing Strategies: A Survey. *Software Testing,Verification and Reliability* 15, 3 (2005), 167–199.

[3] J.D. Hagar, T.L. Wissink, D.R. Kuhn, and R.N. Kacker. 2015. Introducing combinatotial testing in a large organization. 48, 4 (2015), 64–72.

[4] D.R. Kuhn, R.N. Kacker, and Y. Lei. 2010. Practical Combinatorial testing. *NIST special Publication* 800, 142 (2010).

[5] D.R. Kuhn and M.J. Reilly. 2002. An investigation of the applicability of design of experiments to software testing. In *Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW '02)*. 91f–95.

[6] D.R. Kuhn, D.R. Wallace, and G. J. AM. 2004. Software fault interactions and implications for software testing. 30, 6 (2004), 418–421.

[7] R. Kuhn, R. kacker, Y. Lei, and J. Hunter. 2009. Combinatorial software testing. 42, 8 (2009), 94–96.

[8] C. Nie and H. Leung. 2011. A survey of combinatorial testing. *Comput. Surveys* 43, 2 (2011), 11.

[9] C. Nie, H. Leung, and K-Y. Cai. 2013. Adaptive combinatorial testing. In *2013 13th international conference on quality software*. 284–287.

[10] S. Vilkomir, O. Starov, and R. Bhambroo. 2013. Evaluation of t-wise approach for testing logical expressions in software. In *Proceedings of the 6th IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW '13)*. IEEE, 249–256.

[11] Z. Zhang and J. Zhang. 2011. Characterizing failure-causing parameter interactions by adaptive testing. In *Proceedings of the 20th International Symposium on Software Testing and Analysis (ISSTA '11)*. ACM, 331–341.

ACM Permission and Release Form

Title of non-ACM work: Poster: A Weight Approach to Combinatorial Test Generation
Author(s): Jing Zhao:Dalian University of Technology;Gao Rong Ning:CSIC BeiJing; H.L.LU, Y.B.Wang, Yan Cai, Jian Zhang

Type of material: **Poster; supplemental material(s)**

TITLE OF ACM PUBLICATION: 40th International Conference on Software Engineering Companion Proceedings

Grant Permission

As the owner or authorized agent of the copyright owner(s) I hereby grant non-exclusive permission for ACM to include the above-named material (the *Material*) in any and all forms, in the above-named publication.

I further grant permission for ACM to distribute or sell this submission as part of the above-named publication in electronic form, and as part of the ACM Digital Library, compilation media (CD, DVD, USB) or broadcast, cablecast, laserdisc, multimedia or any other media format now or hereafter known. (*Not all forms of media will be utilized.*)

☑ Yes, I grant permission as stated above.

Multiple Author Submission Options
◉ I am submitting this permission and release form on behalf of all co-authors
○ I cannot submit this permission and release form on behalf of all co-authors

The following notice of publication and ownership will be displayed with the Material in all publication formats:

The new ACM Consolidated TeX template Version 1.3 and above automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\copyrightyear{2018}
\acmYear{2018}
\setcopyright{rightsretained}
\acmConference[ICSE '18 Companion]{40th International Conference on Software Engineering Companion}{May 27-June 3, 2018}{Gothenburg, Sweden}
\acmBooktitle{ICSE '18 Companion: 40th International Conference on Software Engineering Companion, May 27-June 3, 2018, Gothenburg, Sweden}\acmDOI{10.1145/3183440.3195018}
```

\acmISBN{978-1-4503-5663-3/18/05}

ACM TeX template .cls version 2.8, automatically creates and positions these text blocks for you based on the code snippet which is system-generated based on your rights management choice and this particular conference.

Please copy and paste the following code snippet into your TeX file between \begin{document} and \maketitle, either after or before CCS codes.

```
\CopyrightYear{2018}
\setcopyright{rightsretained}
\conferenceinfo{ICSE '18 Companion}{May 27-June 3, 2018, Gothenburg,
Sweden}\isbn{978-1-4503-5663-3/18/05}
\doi{https://doi.org/10.1145/3183440.3195018}
```

If you are using the ACM Microsoft Word template, or still using an older version of the ACM TeX template, or the current versions of the ACM SIGCHI, SIGGRAPH, or SIGPLAN TeX templates, you must copy and paste the following text block into your document as per the instructions provided with the templates you are using:

Audio/Video Release
* Your Audio/Video Release is conditional upon you agreeing to the terms set out below.

I further grant permission for ACM to record and/or transcribe and reproduce my presentation and likeness in the conference publication and as part of the ACM Digital Library and to distribute the same for sale in complete or partial form as part of an ACM product on CD-ROM, DVD, webcast, USB device, streaming video or any other media format now or hereafter known.

I understand that my presentation will not be sold separately as a stand-alone product without my direct consent. Accordingly, I further grant permission for ACM to include my name, likeness, presentation and comments and any biographical material submitted by me in connection with the conference and/or publication, whether used in excerpts or in full, for distribution described above and for any associated advertising or exhibition.

Do you agree to the recording, transcription and distribution? ◉ Yes ○ No

Auxiliary Materials, not integral to the Work

Do you have any Auxiliary Materials? ○ Yes ◉ No

Third Party Materials * http://www.acm.org/publications/third-party-material

In the event that any materials used in my submission or Auxiliary Materials contain the work of third-party individuals or organizations (including copyrighted music or movie excerpts or anything not owned by me), I understand that it is my responsibility to secure any necessary permissions and/or licenses for print and/or digital publication, and cite or attach them below. Third-party copyright must be clearly stated in the caption(s) or images or in the text narrative near the object(s) in the Work and in any presentation of it and in Auxiliary Materials as applicable.

ACM offers Fair Use Guidelines at
http://www.acm.org/publications/guidance-for-authors-on-fair-use

* Small-performing rights licenses must be secured for the public performance of any copyrighted musical composition. Synchronization licenses must be secured to include any copyrighted musical composition in film or video presentations.

◉ I have not used third-party material.

○ I have used third-party materials and have necessary permissions.

Representations, Warranties and Covenants

The undersigned hereby represents, warrants and covenants as follows:

(a) Owner is the sole owner or authorized agent of Owner(s) of the Work;

(b) The undersigned is authorized to enter into this Agreement and grant the rights included in this license to ACM;

(c) The Work is original and does not infringe the rights of any third party; all permissions for use of third-party materials consistent in scope and duration with the rights granted to ACM have been obtained, copies of such permissions have been provided to ACM, and the Work as submitted to ACM clearly and accurately indicates the credit to the proprietors of any such third-party materials (including any applicable copyright notice), or will be revised to indicate such credit.

(d) The Work has not been published except for informal postings on non-peer reviewed servers, and Owner covenants to use best efforts to place ACM DOI pointers on any such prior postings;

(e) The Auxiliary Materials, if any, contain no malicious code, virus, trojan horse or other software routines or hardware components designed to permit unauthorized access or to disable, erase or otherwise harm any computer systems or software; and

(f) The Artistic Images, if any, are clearly and accurately noted as such (including any applicable copyright notice) in the Submitted Version.

Additionally, please reference the following representations that must be agreed to prior to submission and acceptance of your paper.

http://www.acm.org/publications/policies/author_representations

☑ I agree to the Representations, Warranties and Covenants.

DATE: **03/07/2018** sent to zhaoj9988@126.com at **06:03:14**