An Efficient Algorithm to Identify Minimal Failure-Causing Schemas from Exhaustive Test Suite

Yuanchao Qi[1], Qi Wang[1], Chiya Xu[1], Tieke He[2], and Ziyuan Wang[1*]
[1]School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, China
[2]State Key Lab for Novel Software Technology, Nanjing University, Nanjing, China
*Corresponding: wangziyuan@njupt.edu.cn

***Abstract*—Combinatorial testing is widely used to detect failures caused by interactions among parameters for its efficiency and effectiveness. Fault localization plays an important role in this testing technique. And minimal failure-causing schema is the root cause of failure. In this paper, an efficient algorithm, which identifies minimal failure-causing schemas from existing failed test cases and passed test cases, is proposed to replace the basic algorithm with worse time performance. Time complexity of basic and improved algorithms is calculated and compared. The result shows that the method that utilizes the differences between failed test cases and passed test cases is better than the method that only uses the sub-schemas of those test cases.**

***Keywords*—Combinatorial testing, fault localization, minimal failure-causing schema, algorithm.**

I. Introduction

Softwares may be affected by the interactions among its parameters. These interactions need to be tested to guarantee the quality of software. But for the software with k parameters, it is unacceptable to cover all the possible k-tuple combinations of parametric values. Combinatorial testing provides a trade-off between the testing cost and the degree of combinatorial coverage. It has been widely used for its efficiency and effectiveness, especially, in highly-configurable systems [1]. Test case generation and fault localization based on failure-causing schemas are hot areas of research.

Many researches focus primarily on generating test cases to filter suspicious failure-causing schemas. But there are few materials to discuss how to filter them. The default option is to construct all possible suspicious failure-causing schemas from failed test cases and filter them by passed test cases. In this paper, we propose an improved algorithm with better time performance. The analysis of time complexity shows the advantage of proposed algorithm. And the efficient algorithm has been applied in our practice to compute minimal failure-causing schemas for boolean-specification testing and Siemens program suite.

II. Background

The model of minimal failure-causing schema was proposed by Nie et. al [2].

For a software system with k parameters, we suppose each parameter f_i has a_i $(1 \leq i \leq k)$ discrete valid values. Let $F = \{f_1, f_2, ..., f_k\}$ denote the set of parameter, and $V_i = \{0, 1, ..., a_{i-1}\}$ $(i = 1, 2, ...k)$ the value set for f_i without loss of generality.

Definition 1. (Schema). A k-tuple s=(-, ...,- ,v_{i_1}, -,..., -, v_{i_2}, -, ..., -, v_{i_τ}, -, ..., -) is a schema with strength τ, or a τ-way schema (or τ-schema for short) $(1 \leq \tau \leq k)$. Where τ values are fixed as $v_{i_1} \in V_{i_1}$, $v_{i_2} \in V_{i_2}$, ..., $v_{i_\tau} \in V_{i_\tau}$, and other $k-\tau$ values are not fixed and represented as "-".

Definition 2. (Sub-schema and parent-schema). Schemas $s_1 = (v_1, v_2, ..., v_k)$ and $s_2 = (v'_1, v'_2, ..., v'_k)$ are τ_1-schema and τ_2-schema respectively $(\tau_1 \leq \tau_2)$. If $\forall 1 \leq i \leq k$, (v_i=-) $\vee$ ($v_i = v'_i$) is true, then s_1 is a sub-schema of s_2, and s_2 is a parent-schema of s_1. It is denoted as $s_1 \prec s_2$. Especially, if $s_1 \neq s_2$, then s_1 is a real sub-schema of s_2, and s_2 is a real parent-schema of s_1.

Definition 3. (Failure-causing schema). A schema s is a failure-causing schema (or FS for short), if $\forall t \in T_{all} = V_1 \times V_2 \times ... \times V_k$, $s \prec t \Rightarrow t$ is failed test case.

Definition 4. (Minimal failure-causing schema). A failure-causing schema s is a minimal failure-causing schema (or MFS for short), if any real sub-schema of s is not a failure-causing schema.

III. Algorithms

People pay more attention to the problem that how to identify minimal failure-causing schemas accurately. However, efficiency is also a fundamental issue. In this section, we will introduce the most used basic algorithm and then propose an improved one with better time complexity.

A. Basic algorithm

The basic algorithm, which identifies minimal failure-causing schemas from failed test cases and passed test cases, was mentioned in many materials. But its detailed process was often omitted. Here we describe the basic algorithm and analyze its time performance.

For each failed test case t, there are C_k^1 1-way sub-schemas, C_k^2 2-way sub-schemas,..., and C_k^k k-way schemas.

When filtering suspicious schemas, from $nf \times C_k^1$ 1-way sub-schemas of failed test cases will be filtered by $np \times C_k^1$ 1-way sub-schemas of passed test cases, to $nf \times C_k^k$ k-way sub-schemas of failed test cases be filtered by $np \times C_k^k$ k-way sub-schemas.

When comparing two i-way schemas $(i = 1, 2, ..., k)$, the values of i parameters should be compared. Therefore, the total

DOI reference number: 10.18293/SEKE2016-233

Algorithm 1: Identify MFS using failed and passed test cases
Input: $FTCS$: set of failed test cases
$PTCS$: set of passed test cases
Output: $MFSs$: set of minimal failure-causing schemas
1. $FSs = \emptyset$;
2. **For Each** failed test case $t \in FTCS$
3. $\quad FSs = FSs + SubScheSet(t)$;
4. **End For**
5. **For Each** passed test case $t \in PTCS$
6. $\quad FSs = FSs - SubScheSet(t)$;
7. **End For**
8. $MFSs = \{s | s$ is minimal schemas in $FSs\}$;

time complexity of filtering suspicious schemas in Algorithm 1 should be:

$$O(np \times C_k^1 \times nf \times C_k^1 \times 1 + np \times C_k^2 \times nf \times C_k^2 \times 2 + ... + np \times C_k^k \times nf \times C_k^k \times k) \sim O(np \times nf \times \sum_{i=1}^{k}(i \times (C_k^i)^2)).$$

Additionally, in the process of selecting minimal ones from the set of failure-causing schemas, we can filter τ-way failure-causing schemas by $(\tau-1)$-way's $(\tau = 2, 3, ..., k)$, for each failed test case. So there are totally $C_k^k \times C_k^{k-1} + C_k^{k-1} \times C_k^{k-2} + ... + C_k^{k-2} \times C_k^1$ parametric values should be checked for each failed test case. Here note that $O(\sum_{i=2}^{k}(C_k^i \times C_k i - 1)) \sim O(\sum_{i=2}^{k}(C_k^i)^2) \sim O(\sum_{i=1}^{k}(C_k^i)^2) \sim O(C_{2k}^k)$.

Therefore, the time complexity of the whole Algorithm 1 should be: $O(np \times nf \times \sum_{i=1}^{k}(i \times (C_k^i)^2) + nf \times C_{2k}^k)$.

B. Improved algorithm

Factually, the process of extracting and filtering suspicious schemas in the basic algorithm could be optimized to enhance its time performance. We will propose an improved algorithm by utilizing the differences between failed test cases and passed test cases.

Considering a failed test case t and a passed test case t', we could construct a set of parameteres $Diff\_Param(t, t')$ that contains all parameters whose parametric values in t and t' are different. So, the process to identify MFS is described in Algorithm 2.

Algorithm 2: Identify MFS using failed and passed test cases
Input: $FTCS$: set of failed test cases
$PTCS$: set of passed test cases
Output: $MFSs$: set of minimal failure-causing schemas
1. $FSs = \emptyset$;
2. **For Each** failed test case $t \in FTCS$
3. $\quad Diff(t) = \emptyset$;
4. $\quad$ **For Each** passed test case $t' \in PTCS$
5. $\quad\quad Diff(t, t') = \{f_i \in F | t[i] \neq t'[i]\}$;
6. $\quad\quad Diff(t) = Diff(t) + \{Diff(t, t')\}$;
7. $\quad$ **End For**
8. $\quad FSs(t) = \{s \in SubScheSet(t) |$
. $\quad\quad$ for each $Diff(t, t') \in Diff(t), \exists f_i \in$
. $\quad\quad Diff(t, t')$ that $s[i] \neq$ -$\}$;
9. $\quad FSs = FSs + FSs(t)$;
10. **End For**
11. $MFSs = \{s | s$ is minimal schemas in $FSs\}$;

For a failed test case t, there are k parametric values which should be checked when constructing a $Diff\_Param(t, t')$ with the passed test case t'. If there are np passed test cases, it is $np \times k$. So there are totally $np \times nf \times k$ parametric values which should be checked when constructing these sets for all failed test cases.

For a failed test case t, when selecting its failure-causing sub-schemas, there are np different $Diff\_Param(t, t')$ to be checked. So there are totally $np \times \sum_{i=1}^{k}(i \times C_k^i)$ parametric values which should be checked for one failed test case, and totally $np \times nf \times \sum_{i=1}^{k}(i \times C_k^i)$ parametric values should be checked for all failed test cases.

Therefore, the total time complexity of selecting all failure-causing schemas in Algorithm 2 should be:

$$O(np \times nf \times k + np \times nf \times \sum_{i=1}^{k}(i \times C_k^i))$$
$$\sim O(np \times nf \times \sum_{i=1}^{k}(i \times C_k^i))$$

Since the time complexity of selecting minimal failure-causing schemas is $O(C_{2k}^k)$ for each failed test case, the time complexity of the whole Algorithm 2 should be: $O(np \times nf \times \sum_{i=1}^{k}(i \times C_k^i) + nf \times C_{2k}^k)$.

IV. Discussion

A. Outputs of Two Algorithms

Algorithm 1 and Algorithm 2 obtain the same outputs for the same inputs. It is clear in the description of two algorithms, especially in the description of improved one.

B. Comparing Time Performance

According to the binomial theorem, $C_{2k}^k = \sum_{i=1}^{k}(C_k^i)^2$. Since there are:

$$\sum_{i=1}^{k}(C_k^i)^2 < \sum_{i=1}^{k}(i \times (C_k^i)^2)$$
$$\sum_{i=1}^{k}(i \times C_k^i) < \sum_{i=1}^{k}(i \times (C_k^i)^2)$$

So it is obvious that $O(np \times nf \times \sum_{i=1}^{k}(i \times C_k^i) + nf \times C_{2k}^k) < O(np \times nf \times \sum_{i=1}^{k}(i \times (C_k^i)^2) + nf \times C_{2k}^k)$. It is means that the time complexity of Algorithm 2 is less than that of Algorithm 1. Then we can make a conclusion that Algorithm 2 is better than Algorithm 1.

V. Conclusion

In this paper, we carefully study two algorithms that could identify MFSs by utilizing failed test cases and passed test cases. The time complexity of two algorithms shows that the approach which utilizes the difference between every failed test case and passed test case is clearly better than the other one. We believe that the research will improve the effectiveness and efficiency of practical testing.

References

[1] C. Nie, H. Leung. A survey of combinatorial testing. ACM Computing Surveys (CSUR), 2011, 43(2): 11.

[2] C. Nie, H. Leung. The Minimal Failure-causing Schema of Combinatorial Testing. ACM Transactions on Software Engineering and Methodology (TOSEM), 2011, 20(4): 15.