Journal of Nature Inspired Computing (JNIC)
Vol. 1, No. 2, 2013
Copyright © World Science Publisher, United States
www.worldsciencepublisher.org

A Proposed Model for Minimization of Test Suite

[1] **Ankur Prakash Mudgal**

[1]Amity University Rajasthan, Jaipur, India

Email: ankurmudgal5353@gmail.com

Abstract – Software Testing is a critical part of software development. Software testing and retesting occurs continuously during the software development lifecycle. As software grows and evolves, new test cases are generated and added to a test suite to exercise the latest modifications to the software. Over several versions of the development of the software, some test cases in the test suite may become redundant with respect to the testing requirements for which they were generated since these requirements are now also satisfied by other test cases in the suite that were added to cover modifications in the later versions of software. Due to time and resource constraints for retesting the software every time it is modified, it is important to develop techniques that keep test suite sizes manageable by periodically removing redundant test cases. This process is called test suite minimization. Test suite reduction (TSR) is to find a subset of the test suite containing a minimal number of test cases that can satisfy all test requirements. Test suite reduction techniques attempt to remove redundant test cases. This paper proposes a new model for the minimization of test suite, which is based on the boolean function simplification.

Keywords – Software testing, test case, test suite minimization, regression testing, PMMTS algorithm.

1. Introduction

Before testing a program, testers have to establish testing objectives. A testing objective is considered as a set of testing requirements. Once the set of testing requirements has been determined, test cases are designed to collectively satisfy the testing requirements.

A set of test cases that can collectively satisfy all testing requirements is referred to be a test suite. Test suite development is an expensive process and additionally even conscientiously maintained test suite can grow quite large. Most of the times running an entire suite is not possible as it takes significant amount of time to run all tests in a test suite. So researchers have given various techniques for minimizing test suite. Test suite minimization techniques lower costs by reducing a test suite to a minimal subset that maintains equivalent coverage of original set.

2. Test Suite Reduction

Test suite reduction aims at finding a minimal subset of the test suite that can cover all requirements. It can be stated as follows [1]:
Given: A set of testing requirements R = {r1, r2, …, ri,…, rm} that must be satisfied to provide the desired test coverage of the program, and a set of subsets {T1, T2, …,Tm} of a test suite T = {*t*1, *t*2, …, *t*i, …, *t*n}, one associated with each of the *r*i's such that $\forall$ *t*j $\in$ Ti covers *r*i.
Problem: Find a minimal cardinality subset of T that exercises all *r*i's exercised by the non-minimized test suite T

So far, many algorithms have been used for test suite reduction [1-7], such as, heuristic Algorithm, greedy algorithm and integer programming algorithm.

Metaheuristics algorithms are also applied to various software development phases [8-12].

Harrold et al. propose the heuristic algorithm H to reduce the size of a test suite [6]. The intuition of H is to select test cases according to their "essentialness". H algorithm first groups the requirements r1, r2, …, rm into R1, …, Ri, …, Rd, where Ri (i = 1, …, d) denotes the set of all requirements that are satisfied exactly by i test cases in T, and d is the maximum number of test cases that a requirement can be satisfied. Intuitively speaking, test cases that satisfy requirements in Ri are more "essential" than those that satisfy requirements in Rj. Obviously, test cases that satisfy requirements in R1 are essential. Heuristic H starts by selecting test cases to satisfy requirements in R1 and removes those requirements satisfied by the selected test cases in R. Then, it repeatedly selects the test case that can satisfy a maximum number of unsatisfied requirements in R2 and removes the requirements satisfied by the selected test case in R, until all requirements in R2 are satisfied. Repeatedly, it selects test cases for R3, R4 and so on.

The greedy heuristic algorithm G repeatedly selects the test case in T that satisfies the maximum number of unsatisfied requirements in R [1].

Chen and Lau have proposed a heuristics GE and GRE algorithms [2, 3]. Chen and Lau propose two dividing strategies, i.e. the essentials strategy and 1-to-1 redundancy strategy [5]. Each guarantees the construction of the optimal representative sets of the original problem from those of the sub-problems and the two strategies can be alternately applied. Moreover, Chen and Lau report a simulation study on how often an optimal representative set can be found by means of only the essentials and 1-to-1 redundancy strategies [4].

In general, finding the optimal representative set is equivalent to solving the set-covering problem that is NP-complete[7]. It is the same as the minimization problem of Boolean functions. Both of them can be classified as set-covering problems.

This paper proposes a model which is inspired from the Boolean function simplification to solve the optimal representative set selection problem.

3. Proposed model for minimization of test suite (PMMTS)

3.1 Related concepts

The number of requirements in R may be finite or infinite. However, from a pragmatic point of view, we assume that R is finite, and for each requirement $r \in$ R, there is a test case in the input domain that satisfies *r*. As a result, a finite test suite T always exists. We use *m* and *n* to denote the size of R and T, respectively.

An $n \times (m+1)$ Boolean matrix $A = [a_{ij}]$ is used to describe the satisfaction relation between $\forall\ t_i \in$ T and $\forall$ $r_j \in$ R.

$$\text{Where } a_{ij} = \begin{cases} 0 & t_i \text{ cannot test } r_j \\ 1 & t_i \text{ can test } r_j \end{cases}$$

for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. $a_i(m) = i$. $a_i = (a_{i1}a_{i2} \ldots a_{im})$ is the *i*-th row vector, $b_j = (a_{1j}a_{2j} \ldots\ldots a_{nj})^T$ is the *j*-th column vector. $C1(a_i)$ is the count of "1" in a_i and $C1(b_j)$ is the count of "1" in b_j. The axioms follows in Boolean algebra can also be used here.

Example 1: It is assumed that R = {$r_1, r_2, \ldots, r_8$} and T = {$t_1, t_2, \ldots, t_7$}. A relationship between eight test requirements and seven test cases are shown in Table 1. It leads to a 7 × 9 matrix $\boldsymbol{A} = [a_{ij}]$.

Table.1 Relationship Chart of Example 1

Test Requirements ri	Test Cases Ti
r1	t1, t5
r2	t5
r3	t 1, t 2, t 3
r4	t 3, t 6
r5	t 1, t 4
r6	t 1, t 6
r7	t 3, t 4, t 7
r8	t2, t 3, t 4, t 7

$$\begin{matrix} 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 3 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 4 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 7 \end{matrix}$$

$$A = \begin{pmatrix} \\ \\ \\ \end{pmatrix}$$

Because test cases $t_1, t_2, \ldots, t_n$ are mutually independent, so we can construct the prime implicant chart of example 1 in Table 2. This chart is named as covering chart and can be expressed by matrix $\boldsymbol{A}$.

Table 2. Covering Chart of Example 1

	r1	r2	r3	r4	r5	r6	r7	r8
t1	1	0	1	0	1	1	0	0
t2	0	0	1	0	0	0	0	1
t3	0	0	1	1	0	0	1	1
t4	0	0	0	0	1	0	1	1
t5	1	1	0	0	0	0	0	0
t6	0	0	0	1	0	1	0	0
t7	0	0	0	0	0	0	1	1

Then, some rules as follows will be applied to reduce the chart until the optimal representative set been found.

Rule 1: For $\forall\ \boldsymbol{a}_i \in \boldsymbol{A}$, if each "1" in $\boldsymbol{a}_i$ can be completely contained by other $\boldsymbol{a}_j \in \boldsymbol{A}$ for $j \neq i$, then $\boldsymbol{a}_i$ can be removed from the chart without affecting the completeness of the test suite. It means that for $\boldsymbol{a}_i$, if

$$\overline{a_i} + \sum_{j \neq i} a_j = (111\ldots1)$$

, then $\boldsymbol{a}_i$ can be removed from the chart

Rule 2: During application of Rule 1, if ∃ ***a*i**, ***a*k**, they satisfy that if

$$\overline{a_i} + \sum_{j \neq i} a_j = (111...1)$$
,
$$\overline{a_k} + \sum_{j \neq k} a_j = (111....1)$$
and C1(***a*i**) ≤ C1(***a*k**), then ***a*i** can be removed from the chart.

Rule 3: For ∀ ***b*i** ∈ ***A***, if each "1" in ***b*i** can be completely contained by other ***b*j** ∈ ***A*** for j ≠ i, then ***b*j** can be removed from the chart without changing the completeness of the test suite. It means that

$$\overline{b_i} + b_j = (1\,1\,1\ldots1)^T$$
,
then ***b*j** can be removed from the chart.

Rule 4: The use of Rule 1, Rule 2 and Rule 3 repeatedly will not affect the completeness of the test suite.
The maintenance of the completeness of the test suite is explained briefly by the following example.
Example 2: Consider ***a*2** ∈ ***A*** in Example 1, where ***a*2** = (00100001), it means that the test case *t*2 can test *r*3 and *r*8. However, *r*3 and *r*8 can also be tested by *t*3, so {*r*3, *r*8} can also be test by T' = T - {*t*2}. It results *t*2 to be a redundant test case and can be removed from T.

Consider ***b*2** ∈ ***A*** in Example 1, where $b_2 = (0000100)^T$, it means that {*t*5} can test *r*2. For ***b*1** ∈ ***A***, where $b_1 = (1000100)^T$, {*t*1, *t*5} can test *r*1. Thus, *t*5 can test both *r*2 and *r*1. Therefore, if *r*1 is removed from the covering chart, the completeness of the test suite will not change.

3.2 Algorithm on proposed method for minimization of test suite (PMMTS)

Step 1. Construct matrix *A* based on the relationships between test case set T and test requirement set R;
Step 2. Calculate C1(*a*i);
Step 3. *A*→*A'* by applying Rule 1 and Rule 2;
Step 4. Calculate C1(*b*i);
Step 5. *A'*→*A"* by applying Rule 3. If there is more than one *b*i, all of them should be removed;
Step 6. Loop Step 2 - Step 5 until no redundant *a*i or *b*i in matrix;
Step 7. Print the obtained matrix *A"*;
Step 8. Print *a*i(m).

The matrix transformation given below shows the solution procedure of Example1 by using the PMMTS algorithm

$$\begin{pmatrix} 1&0&1&0&1&1&0&0&1\\ 0&0&1&0&0&0&0&1&2\\ 0&0&1&1&0&0&1&1&3\\ 0&0&0&0&1&0&1&1&4\\ 1&1&0&0&0&0&0&0&5\\ 0&0&0&1&0&1&0&0&6\\ 0&0&0&0&0&0&1&1&7 \end{pmatrix} \xrightarrow{\text{remove row}} \begin{pmatrix} 1&0&1&0&1&1&0&0&1\\ 0&0&1&1&0&0&1&1&3\\ 1&1&0&0&0&0&0&0&5 \end{pmatrix}$$

$$\begin{pmatrix} 0&0&1&1\\ 0&1&0&3\\ 1&0&0&5 \end{pmatrix}$$

So far, ***A*** changes to an identity matrix if the last column is ignored.
The optimal representative set of Example 1 is {*t*1, *t*3, *t*5}.

4. Results

We have considered 10 different problems with different no. of requirements and their different satisfying test cases. The following table and graph describes about the reduction in no. of test cases on applying our PMMTS algorithm. There is maximum of 66.6% reduction in "B" and minimum of 42.8% in "G".

Table.3 Containing Data of 10 Problems

Problem	Original No. of Test cases	Reduced No. of Test Cases	Reduction in Test Cases (in %)
A	9	4	55.5
B	9	3	66.6
C	9	4	55.5
D	8	3	62.5
E	8	4	50
F	8	4	50
G	7	4	42.8
H	7	3	55.5
I	6	4	33.3
J	6	3	50

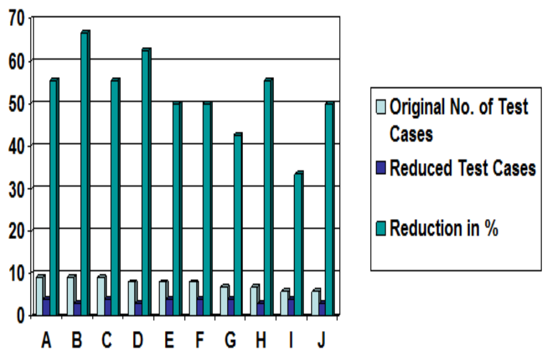

Figure 1: Graphical representation of Table 3.

5. Discussion

The reduction of test suite is to find an optimal representative set, which means to find a minimal subset from a given set to satisfy given requirements. Both the minimization of Boolean functions and the reduction of test suite are equivalent to solving the set covering problem. So the methods to solve the minimization of Boolean functions problem can also be adapted to solve the reduction of test suite problem.

6. Conclusions

The PMMTS algorithm given in the paper can be used at testing and maintenance phases of software development.The algorithm has been implemented in java language. The cost of implementing the algorithm is almost negligible but it saves the cost and effort of running extra test cases. the results of experiments show its certain advantages.

References

[1] M.J. Harrold, R. Gupta, and M.L. Soffa, "A methodology for controlling the size of a test suite", ACM Transactions on Software Engineering and Methodology, Vol. 2, No. 3, 1993,pp. 270−285.

[2] A.J. James and J.H. Mary, "Test-Suite Reduction and Prioritization for Modified Condition and Decision Coverage",IEEE Transactions on Software Engineering, Vol. 29, No. 3,2003, pp. 195-209.

[3] T.Y. Chen and M.F. Lau, "A new heuristic for test suite reduction", Information and Software Technology, Vol. 40, Nos. 5-6, 1998, pp. 347–354

[4] T.Y. Chen and M.F. Lau, "Dividing strategies for the optimization of a test suite", Information Processing Letters,Vol. 60, No. 3, 1996, pp. 135–141.

[5] T.Y. Chen and M.F. Lau, "Heuristics towards the optimization of the size of a test suite", Proceedings of the Third International Conference on Software Quality Management, Seville, 1995, pp. 415–424.

[6] T.Y. Chen and M.F. Lau, "On the divide-and-conquer approach towards test suite reduction", *Information Science*,Vol. 152, 2003, pp. 89−119

[7] J.G. Lee and C.G. Chung, "An optimal representative set selection method", Information and Software Technology, Vol. 42, No. 1, 2000, pp. 17−25.

[8] Tarun Kumar Sharma and Millie Pant, "Enhancing the Food Locations in an Artificial Bee Colony Algorithm", Soft Computing, Springer

[9] Tarun Kumar Sharma and Millie Pant, "Halton Based Initial Distribution in Artificial Bee Colony Algorithm and its Application in Software Effort Estimation", International Journal of Natural Computing Research (IJNCR), IGI Publication, USA, vol. 3(2), pp. 86 - 106, 2012. (DOI: 10.4018/jncr.2012040105).

[10] Tarun Kumar Sharma, Millie Pant, V.P.Singh, Improved Local Search in Artificial Bee Colony using Golden Section Search, Journal of Engineering (JOE) Vol. 1, No. 1, 2012, pp. 14-19.

[11] Tarun Kumar Sharma, Millie Pant and V.P. Singh, "Adaptive Bee Colony in an Artificial Bee Colony for Solving Engineering Design Problems". Advances in Computational Mathematics and its Applications (ACMA) Vol. 1 No.4, 2012, pp. 213 - 221.

[12] Tarun Kumar Sharma, Millie Pant, V.P.Singh, Modified Mutation in Differential Evolution Algorithm to Optimize Supply Chain System; Journal of Nature Inspired Computing (JNIC), vol. 1(1), pp. 1-8, 2012.

Ankur Prakash Mudgal done his Masters in Computer Science in 2010. Presently associated with Amity University Rajasthan as a Lecturer in the department of Information Technology. He has three years of experience. His key research areas are Software Testing, Complexity Analysis.